# Nonlinear Model Predictive Control Reduction Strategies for Real-time Optimal Control

by

Anson Maitland

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Engineering

Waterloo, Ontario, Canada, 2019

**Examining Committee Membership**

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

| | |
|---|---|
| External Examiner: | Ya-Jun Pan<br>Professor, Dept. of Mechanical Engineering,<br>Dalhousie University |
| Supervisor: | John McPhee<br>Professor, Dept. of Systems Design Engineering,<br>University of Waterloo |
| Internal Member: | Nasser Lashgarian Azad<br>Associate Professor, Dept. of Systems Design Engineering,<br>University of Waterloo |
| Internal Member: | William Melek<br>Professor, Dept. of Mechanical and Mechatronics Engineering,<br>University of Waterloo |
| Internal-External Member: | Krzysztof Czarnecki<br>Professor, Dept. of Electrical and Computer Engineering,<br>University of Waterloo |

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

This thesis presents a variety of strategies to accelerate the turnaround times (TATs) of nonlinear and hybrid model predictive controllers (MPCs). These strategies are unified by the themes of symbolic computing, nonlinear model reduction and automotive control.

The first contribution of this thesis is a new MPC problem formulation, called symbolic single shooting (symSS), that leverages the power of symbolic computing to generate an optimization problem of minimal dimension. This formulation is counter to the recent trend of introducing and exploiting sparsity of the MPC optimization problem for tailored solvers to exploit. We make use of this formulation widely in this thesis.

The second contribution of this thesis is a novel application of proper orthogonal decomposition (POD) to MPC. In this strategy we construct a dimensionally-reduced optimization problem by restricting the problem Lagrangian to a subspace. This subspace is found by running simulations offline from which we extract the important solution features. Using this restricted Lagrangian we are able to reduce the problem dimension dramatically, thus simplifying the linear solve. This leads to TAT accelerations of more than two times with minimal controller degradation.

The third contribution of this thesis is an informed move blocking strategy. This strategy exploits the features extracted in the restricted Lagrangian subspace to derive a sequence of increasingly blocked move blocking strategies. These move blocking strategies can then be used to reduce the dimension of the optimization problem in a sparse manner, leading to even greater acceleration of the controller TAT .

The fourth contribution of this thesis is a new quasi-Newton method for MPC. This method utilizes ideas similar to singular perturbation-based model reduction to truncate the expression for the problem Hessian at the symbolic level. For nonlinear systems with a modest Lipschitz constant, we can identify the timestep as a 'small' parameter about which we can do a perturbative expansion of the Lagrangian and its derivatives. Truncating to first order in the timestep, we are able to find a good approximation of the Hessian leading to TAT acceleration.

The fifth contribution of this thesis is controller integration strategy based on nested MPCs. Using the symSS formulation we can construct an explicit model of a controlled plant that includes the full model as well as the MPC's action. This form of the controlled plant model allows us to generate exact derivatives so that fast solvers can be used for real time application. We focus here on the problem of planning and motion control integration for autonomous vehicles but this strategy can be extended for other problems that require accurate models of a controlled plant.

The sixth contribution of this thesis is a strategy to handle integer controls in MPC based on a few reasonable assumptions: our predictions over the horizon are *almost* perfect and the future is inevitable. These assumptions enforce a degree of continuity, in the integer controls, between solutions over different timesteps that allow us to mitigate chatter and enforce a hard upper bound on solution complexity. This strategy constrains the integer solution of one timestep to be related to that of the previous timestep. Our results show that this strategy provides acceptable control performance while achieving TATs that are orders of magnitude smaller than those for conventional MINLP-based methods, thereby opening the door to new real-time applications of hybrid MPC.

**Acknowledgements**

Foremost, I would like to sincerely thank my supervisor Professor John McPhee for his encouragement, empathy and wisdom during my research. I am indebted to his guidance.

I would like to thank my colleagues Dr. Mohit Batra for the development of the electric vehicle cruise control MPC and feedback on the symSS approach and Chris Shum for his troubleshooting help with the MicroAutobox II 1401 real-time computer. I would also like to thank Dr. Behzad Samadi, Dr. Kevin Walker and Dr. Jürgen Gerhard of Maplesoft™ for their helpful advice and Dr. Ken Butts of Toyota for sharing the diesel airpath model and providing useful feedback.

Lastly, I would like to thank my wife for her constant support and patience during the preparation of this thesis.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Maplesoft™.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Model Predictive Control (MPC) is a modern control strategy that promises significant benefits over established control methods. For example, the widespread application of MPC to the automotive world would be revolutionary; however it has been recognized that computational burden of MPC is too great for the electronic control units (ECUs) used in the automotive industry [1]. The computational demands of MPC is the foremost hurdle to its adoption by industry. In its most general form MPC requires the online solution of an optimization problem, which in practise must be completed within a single sampling period, typically on the order of milliseconds or less. Thus, to deploy MPC on a commercial scale and realize its purported benefits we require rapid controller turnaround times (TATs) [2, 3]. This is the motivating problem for this thesis. We present a variety of strategies to address the hard real-time constraint for nonlinear and hybrid MPC.

There are a few themes that unify this thesis: nonlinear model reduction, symbolic computing and automotive control. In the following chapters one or more of these themes will be present. Model reduction is the application of mathematical tools and heuristics to approximate a model with a computationally simpler and/or lower-order model. It is a well-known strategy to reduce the TAT of an MPC. Since an MPC requires a model to predict the plant's future states, by reducing the order and/or complexity of the plant model significant computational savings can be had. Reduction methods for linear models, such as Balanced Truncation and Proper Orthogonal Decomposition (POD), are the most well developed reduction techniques [4]. They are mature methods with a wealth of literature on their applications. These methods operate by reducing the order of the model by capturing only the most important states through a linear coordinate change. They have also been applied, with some success, to nonlinear systems. Other reduction methods for nonlinear systems include trajectory piecewise linearization, which combines order reduction with the

practise of model linearization common to controls [5] and realisation-preserving methods, such as importance analysis [6] and activity indices [7].

This work began as a search for nonlinear model reduction methods oriented towards modern control applications and reduced TATs. In our search we found that, with some careful modifications, model reduction methods can be applied directly to the controller. In particular, we present a new application of POD to optimal control in Chapter 4. By restricting the Lagrangian of the finite horizon optimal control problem (FHOCP) of an MPC to a suitable affine subspace, we can achieve a reduction in computational cost leading to faster TATs with minimal degradation in controller performance. Unlike many of the existing controller reduction strategies, that focus on the solver, the POD-based method presented here is applied at the level of the problem statement (the FHOCP Lagrangian). The results of this method are very promising and lead to a new move-blocking generation strategy, as well. The informed move-blocking strategy we present in Section 4.3 increases the sparsity of the reduced MPC and accelerates TATs further.

Symbolic computing tools utilize software that is capable of manipulating mathematical expressions and objects. This allows users to do many things traditional software tools cannot do, such as compute derivatives symbolically and carry out algebraic simplifications on mathematical expressions. We make extensive use of symbolic computing tools (primarily Maple) to generate fast optimized code for online optimization. The advantages of symbolics in reducing computation time has been known for some time in the world of dynamics and simulation [8]. In particular, we fully leverage the power of symbolic computing and present a new formulation of the FHOCP called symbolic single shooting (symSS) in Chapter 3, where only the control inputs, and not the states, are exposed as degrees of freedom in the transcribed optimization problem. This strategy doesn't readily fit into the taxonomy of existing FHOCP formulations but yields a denser optimization problem as compared to direct collocation, the most popular approach, which treats plant dynamics as equality constraints.

The symSS approach shares the same degrees of freedom as the traditional single shooting method. In its standard implementation, single shooting has two stages. The model is first simulated in an inner loop and then the derivatives are computed, through sensitivity or adjoint equations, for the optimizer that runs in an outer loop where the control values are updated. The symSS approach generates a controller that operates efficiently in a single stage. In Chapter 5 we extend this approach to generate a new quasi-Newton method that uses a computationally reduced Hessian at the symbolic level. By carrying out a straightforward perturbative expansion of the Lagrangian of the FHOCP and its derivatives we are able to acquire a good (under mild assumptions) first order approximation of the expressions for the Hessian.

Automotive control problems are those related in any way to the operation of a vehicle. In this thesis we look at a variety of autonomous vehicle tracking problems, diesel airpath control and electric vehicle longitudinal control. These types of problems are motivating problems for this work, as MPC promises to realize distinct advantages in efficiency and autonomy over existing controllers. However, the aforementioned applications require rapid sampling times on the order of milliseconds or less. A particular problem we frequently consider in this thesis is the dynamic reference tracking problem. The goal of the MPC in this case is to make a vehicle follow a reference path and speed profile. An immediate issue that arises in setting up this problem is the selection of the reference trajectory. One can choose the reference rather arbitrarily without any guarantee of the MPC's success. In a real-world scenario the reference would be generated by some local path planning module. In Chapter 6 we propose a nested MPC approach to integrate the MPC controlled vehicle with local path planning, to try to guarantee the reference is controller feasible. The development of the nested MPC approach is only possible with the symSS formulation and symbolic computing tools. We are able to generate an explicit model of the MPC controlled vehicle that can be integrated with a local planner in real-time.

Furthermore, control of modern vehicles can be complicated by the presence of hybrid operation of the powertrain. In Chapter 7 we present a new strategy to handle the difficulties presented by integer controls in MPC. This strategy uses quasi-translations (QTs) to constrain the set of possible integer control sequences considered at each timestep of an MPC. Effectively, the method enforces a degree of continuity in the implemented integer controls over successive timesteps to mitigate chatter and reduce controller TATs. Like the POD reduced MPC method, this strategy works by taking advantage of the sequential nature of an MPC in order to reduce the computational burden of online optimization. Using the symSS approach with QTs, we are able to handle models with both explicit and implicit hybrid behaviour in real-time.

In summary (of above) the main contributions of this thesis are:

1. the symSS formulation of the FHOCP
2. POD reduced MPC
3. the Informed Move Blocking strategy
4. the truncated Lagrangian method for symSS
5. nested MPC design using symSS for controller integration
6. the QT strategy for mixed-integer MPC

# Chapter 2

# Review of Model Predictive Control

In this chapter we review the basics of MPC. We focus here on its problem formulation and method of solution. MPC can be understood as a tractable method to approximate the dynamic programming (DP) solution of a nonlinear optimal control problem and is related to the classical Linear Quadratic Regulator control. Thus much of the analysis of nonlinear MPC focuses on set point trajectories. Through proper selection of terminal constraints or terminal costs one can approximate the cost-to-go function found in DP in order to provide optimality guarantees. Further, the choice of cost function in combination with these terminal ingredients can be used with Lyapunov theory to ensure controller stability and even recursive feasibility. There exist a number of excellent texts that provide further theoretical details; see [9, 10].

## 2.1 MPC Formulation

MPC is a forward looking, feedback-based control strategy that can solve (in approximation) real-world optimal control problems [1, 11]. An MPC operates under the receding horizon principle by optimizing the controller actions on a model of the plant over some future horizon, implements only the first controller action, and then repeats this process at each timestep. From a theoretical perspective an MPC is simply a method to approximate a dynamic programming solution over a finite horizon.

For simplicity we consider a single stage, time-independent optimal control problem

which is often of the following form

$$\begin{aligned} \min_{\mathbf{x}(t),\mathbf{u}(t)} & \left\{ \mathcal{J}(\mathbf{x}(t),\mathbf{u}(t)) = S(\mathbf{x}(t_0),\mathbf{x}(t_f)) + \int_{t_0}^{t_f} V(\mathbf{x}(t),\mathbf{u}(t))dt \right\} \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \boldsymbol{\phi}(\mathbf{x}(t),\mathbf{u}(t)) \\ & \mathbf{H}(\mathbf{x}(t_0),\mathbf{x}(t_f),\mathbf{x}(t),\mathbf{u}(t)) \leq 0 \end{aligned} \tag{2.1}$$

where time $t \in [t_0, t_f]$ is the independent variable, $\mathbf{x} \in \mathbb{R}^n$ are the system states and $\mathbf{u} \in \mathbb{R}^m$ are the controls. The first constraints are dynamic constraints coming from the differential equations representing the plant model with vector field $\boldsymbol{\phi}$. The constraint vector $\mathbf{H}$ enforces all other constraints, such as path, terminal and control constraints, as well as initial conditions. Problem (2.1) is an infinite-dimensional, open-loop, optimization problem that cannot be readily solved by computer. Hence, the requirement for closed loop strategies, like MPC, that reformulate problem (2.1) into a tractable form that can handle the uncertainty of the real-world. We refer to the optimization problem of MPC that is solved online as a finite horizon optimal control problem (FHOCP).

Broadly speaking there are two approaches to formulating the FHOCP: direct and indirect methods [12, 13, 14]. In this thesis, we focus on the direct approach where the FHOCP is first discretized and then transcribed directly to a constrained finite-dimensional optimization problem. In the indirect approach, which is much less popular, the optimal control problem is transformed into a two point boundary value problem (TPBVP) using methods from the calculus of variations, such as the augmented Hamiltonian approach. The TPBVP is then discretized leading to a finite-dimensional root-finding problem.

We now describe the direct transcription method. First, we discretize the plant model using some timestep $\Delta t$, to yield a discretized plant model $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k);\Delta t)$ where the discrete index $k$ tracks timesteps. We denote the states and controls over a prediction horizon of $H$ steps by $\mathbf{X} = [\mathbf{x}(0),\ldots,\mathbf{x}(H)]$ and $\mathbf{U} = [\mathbf{u}(0),\ldots,\mathbf{u}(H-1)]$, respectively. The length of the prediction horizon is a design variable – too long and the FHOCP becomes too large to meet the real-time constraint, too short and the MPC may fail. One may also select a control horizon $1 \leq H_c \leq H$ separately in order to achieve a balance. In this case the tail of the $\mathbf{U}$ becomes constant, *i.e.* $\mathbf{u}(H_c-1) = \mathbf{u}(H_c) = \ldots = \mathbf{u}(H-1)$. The transcribed FHOCP is then

$$\begin{aligned} \min_{\mathbf{X},\mathbf{U}} & \left\{ \mathcal{J}(\mathbf{X},\mathbf{U};\mathbf{x}(0)) = \ell_H(\mathbf{x}(H)) + \sum_{k=0}^{H-1} \ell_k(\mathbf{X},\mathbf{U}) \right\} \\ \text{subject to} \quad & \mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k);\Delta t), \quad k = 0,\ldots,H-1 \\ & \mathbf{h}(\mathbf{X},\mathbf{U};\mathbf{x}(0)) \leq 0 \end{aligned} \tag{2.2}$$

where $\mathbf{x}(0)$ is the most recent (possibly estimated) state coming from plant feedback, $\ell_k$ are the stage costs with terminal cost $\ell_H$ and the constraint vector $\mathbf{h}$ captures the discretized constraints. We denote the solution of (2.2) by $(\mathbf{X}^*, \mathbf{U}^*)$. Following the receding horizon principle, only $\mathbf{u}^*(0)$ is implemented at the current timestep before $\mathbf{x}(0)$ is refreshed and (2.2) is solved again at the next timestep.

The flexibility of the MPC strategy comes from the generality of the formulation of (2.2) which, in theory, can include arbitrary plant models, constraints and stage costs. However, this flexibility is also a major hurdle in the deployment of MPCs to real-world applications. In particular, the FHOCP of an MPC is a nonlinear programming (NLP) problem which is known to be NP-hard [15, 16]. Given the real-time constraint of a real-world system, the challenge for MPC is increased because not only does an NLP have to be solved at every timestep but it has to be solved within the period of a single timestep. This remains a significant challenge for the deployment of MPC. Often a control engineer must trade accuracy (*e.g.* higher-fidelity models) for computational simplicity in order to meet the real-time demand, reducing the potential benefits of the MPC strategy.

There exist a number of methods to solve NLP (2.2). These are often classified as simultaneous versus sequential methods. Really, they exist along a continuum and differ in how they handle the dynamic constraints. For simplicity of presentation suppose $\mathbf{h} = \emptyset$. The Lagrangian of NLP (2.2) can then be written as

$$\mathcal{L}(\bar{\mathbf{X}}, \mathbf{U}; \tilde{\mathbf{X}}, \mathbf{x}(0)) = \mathcal{J}([\bar{\mathbf{X}}; \tilde{\mathbf{X}}], \mathbf{U}; \mathbf{x}(0)) + \sum_{i \in \mathcal{I}} \boldsymbol{\lambda}(i)[\mathbf{x}(i) - \mathbf{f}(\mathbf{x}(i-1), \mathbf{u}(i-1); \Delta t)]$$

where $\bar{\mathbf{X}} = \{\mathbf{x}(i) | i \in \mathcal{I}\}$, $\tilde{\mathbf{X}} = \mathbf{X} \setminus \bar{\mathbf{X}}$, $\boldsymbol{\lambda}(i)$ are vectors of Lagrange multipliers and $\mathcal{I} \subseteq \{1, 2, \ldots, H\}$. If $\mathcal{I} = \{1, \ldots, H\}$ then $\bar{\mathbf{X}} = \mathbf{X}$ and the method is direct collocation where the dynamics are solved for simultaneously with the controls. If $\mathcal{I} = \emptyset$ then $\bar{\mathbf{X}} = \emptyset$ and the method is single shooting where the states are not optimization variables but parameters of the problem that get updated in an inner integration loop. If $\mathcal{I}$ is neither of the previous options then the method is a multiple shooting method where some states along the horizon are optimization variables to enforce continuity between the smaller sections that are updated within an inner integration loop. The order of the optimization problem is $Hm + 2n|\mathcal{I}|$. From this we can see that the direct collocation forms an NLP of order $H(m + 2n)$ while the NLP of single shooting is of order $Hm$. This difference in dimension is important for solver complexity. The single shooting method provides the smallest problem but comes with the cost of an inner integration loop that can introduce undesirable errors. In chapter 3 we introduce a new approach to single-shooting that takes advantage of symbolic computing to alleviate this issue so we can take advantage of the minimal problem size.

## 2.2 NLP Solvers and MPC

The choice of NLP solver is very important to the implementation of an MPC and many steps have been taken to tailor NLP solvers to FHOCPs of particular structure to reduce the computational burden [17, 18]. A common but not ubiquitous feature of NLP solvers (*e.g.* IPOPT, NPSOL, SNOPT, KNITRO) is the use of Newton's method because of its quadratic convergence rate near the solution. Because Newton's method is so commonly used in derivative-based NLP solvers and because it dominates the computation time of most NLP solvers, we focus on it in this thesis as a way to improve the turnaround times of MPC and ignore many of the other features that may be found in NLP solvers like trust-regions and line searches [19]. In this way it is hoped that many of the results obtained can be implemented within a wide variety of existing NLP solution methods, *e.g.* sequential quadratic programming and interior-point methods.

A general NLP problem can be written as

$$\begin{aligned} &\min_{\mathbf{y}} F(\mathbf{y}) \\ \text{subject to} \quad &\mathbf{g}(\mathbf{y}) = 0 \\ &\mathbf{h}(\mathbf{y}) \leq 0 \end{aligned} \tag{2.3}$$

where $F$ is the objective function and $\mathbf{g} \in \mathbb{R}^p$, $\mathbf{h} \in \mathbb{R}^q$ are constraints. The Lagrangian of this problem is

$$\mathcal{L}(\mathbf{y}) = F(\mathbf{y}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{y}) + \boldsymbol{\nu}^T \mathbf{h}(\mathbf{y})$$

where $\boldsymbol{\lambda} \in R^p, \boldsymbol{\nu} \in \mathbb{R}^q$ are vectors of multipliers. The first-order optimality conditions or Karush-Kuhn-Tucker (KKT) conditions of (2.3) are

$$\begin{aligned} \nabla_{\mathbf{y}} \mathcal{L} &= 0 \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L} &= 0 \quad [\mathbf{g}(\mathbf{y}) = 0] \\ \nabla_{\boldsymbol{\nu}} \mathcal{L} &\leq 0 \quad [\mathbf{h}(\mathbf{y}) \leq 0] \\ \boldsymbol{\nu} &\geq 0 \\ \nu_i h_i(\mathbf{y}) &= 0 \quad i = 1, \ldots, q \end{aligned}$$

In practise this system is not solved directly and there are a multitude of ways to handle the inequality constraints to approximate the problem by a single root-finding problem [19]. For example, one can introduce a barrier function to enforce the inequality constraints.

Using a log barrier (2.3) becomes

$$\begin{aligned} \min_{\mathbf{y}} \left\{ F^{|}(\mathbf{y};\mu) = F(\mathbf{y}) - \mu \sum_{i=1}^{q} \log(-h_i(\mathbf{y})) \right\} \\ \text{subject to} \quad \mathbf{g}(\mathbf{y}) = 0 \end{aligned} \tag{2.4}$$

where $\mu$ is a new parameter. In the limit $\mu \to 0$ the solution of (2.4) approaches the solution of (2.3). The tradeoff to this approach is the numerical ill-conditioning one runs into for small values of $\mu$. The KKT conditions of (2.4) can be captured by the single expression

$$\nabla \mathcal{L}^{|}(\mathbf{z}) = 0$$

where $\mathcal{L}^{|}(\mathbf{z}) = F^{|}(\mathbf{y};\mu) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{y})$ and the optimization variables are $\mathbf{z} = (\mathbf{y}, \boldsymbol{\lambda})$.

Newton's method is a well-known algorithm to find roots of nonlinear equations. A NLP solver often utilizes Newton's method to solve the KKT conditions that arise as the roots of the gradient of a suitably formulated problem Lagrangian $\mathcal{L}$, where the optimization variables (states, controls, multipliers, slack variables, etc.) are given by a single vector $\mathbf{z} \in \mathbb{R}^l$. Simply, Newton's method can be used to solve for $\mathbf{z}^*$ satisfying $\nabla \mathcal{L}(\mathbf{z}^*) = 0$ within an NLP. Newton's method is an iterative technique that repeatedly updates an initial guess $\mathbf{z}^{(0)}$ by taking steps until some termination criterion is reached. The Newton step $\Delta \mathbf{z}_{NS}^{(k)}$ from iterate $\mathbf{z}^{(k)}$ is computed by solving the following linear system

$$\mathbf{H}_{\mathcal{L}}(\mathbf{z}^{(k)}) \Delta \mathbf{z}_{NS}^{(k)} = -\nabla \mathcal{L}(\mathbf{z}^{(k)}) \tag{2.5}$$

where $\mathbf{H}_{\mathcal{L}}$ is the Hessian of $\mathcal{L}$. The next iterate is then computed using the Newton step

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha^{(k)} \Delta \mathbf{z}_{NS}^{(k)} \tag{2.6}$$

where $\alpha^{(k)} \in (0, 1]$ can be found using a line search method. Provided $\mathbf{z}^{(0)}$ is close enough to the solution $\mathbf{z}^*$, repeating steps (2.5) and (2.6) with $\alpha^{(k)} = 1$ will yield a sequence $\{\mathbf{z}^{(k)}\}_{k=0}^{\infty}$ that converges to $\mathbf{z}^*$ quadratically [20].

The computational burden of Newton's method lies in solving (2.5), for which there are two computationally demanding steps: 1. computing the derivatives $\nabla \mathcal{L}$ and $\mathbf{H}_{\mathcal{L}}$ and 2. solving the resultant linear system. Taking advantage of the symmetry of the Hessian, the cost of solving (2.5) is $O(\frac{1}{3} l^3)$ operations, in the worst case, using $\mathrm{LDL}^{\mathrm{T}}$ factorization [20]. Further speedup can be had if the Hessian is sparse, which is the case for some direct transcription methods, *e.g.* direct collocation. In the following chapters of this thesis we present a variety of strategies to alleviate the computational burden of these steps.

# Chapter 3

# Symbolic Single-Shooting for MPC

In this chapter we present a novel formulation of the FHOCP using a discretize then optimize approach via direct transcription. We take advantage of symbolic computing tools to formulate the FHOCP as an unconstrained NLP where only the control inputs are optimization variables. We call this approach symbolic single-shooting (symSS). This formulation is non-standard and doesn't readily fall into the typical direct transcription formulations of direct collocation (DC) or single/multiple-shooting (S/MS) since the dynamic constraints are handled differently [12]. The method is simultaneous, like DC since it doesn't require an inner integration loop, but it eliminates constraints so the resulting NLP is dense unlike those that result from DC. Ultimately, symSS is useful for forming an FHOCP of minimal dimension and for generating optimized code for derivative evaluation that can be utilized in many optimization methods.

The potential advantages of symbolic computing in application to MPC has been recognized in recent years. CasADi, an open source tool for generating derivative information for optimization problems, has only recently matured and been used for a variety of MPC problems [21]. Examples of MPC applications include: control of a subsea pump station [22], control of large scale urban networks [23], Furuta pendulum trajectory optimization [24], UAV trajectory optimization [25] and control of continuous stirred tank reactors [26, 27]. CasADi can carry out automatic differentiation of symbolic expressions to generate derivative information and generate standalone C code, along with a variety of other functions. Its symbolic capabilities are limited, as it is not a full computer algebra system, and it lacks code optimization routines. In these aspects Maple, the symbolic computing software used by the author, has distinct advantages. However, as a tool CasADi is capable of handling the symSS formulation. Of the applications listed above [22, 23, 24, 25] employ

a direct single shooting formulation with CasADi, which is similar to symSS without the use of penalty functions to enforce inequality constraints.

## 3.1 The symSS Formulation

The symSS formulation is as follows. For ease of presentation suppose that the transcribed FHOCP is given by (2.2) where

$$\mathbf{U} \in \{\mathbf{U} | \mathbf{u}(k) \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}] \text{ for } k = 0, \ldots, H-1\}. \tag{3.1}$$

In symSS we rewrite (2.2) with constraints (3.1) as an unconstrained NLP by absorbing the dynamic constraints into explicit recursive expressions inside the cost function and replacing the box constraints with a penalty function to get

$$\min_{\mathbf{U}} \left\{ \mathcal{J}(\mathbf{U}; \mathbf{x}(0)) = \ell_H(\tilde{\mathbf{x}}(H)) + \sum_{k=0}^{H-1} \ell_k(\tilde{\mathbf{X}}, \mathbf{U}) + \|\mathbf{W}\rho_p^2(\mathbf{u}(k))\|_1 \right\} \tag{3.2}$$

where $\tilde{\mathbf{x}}(k) = \mathbf{f} \circ^k (\mathbf{x}(0), \mathbf{U})$ which is defined by the rules

$$\begin{aligned} \mathbf{f} \circ^0 (\mathbf{x}(0), \mathbf{U}) &= \mathbf{x}(0) \\ \mathbf{f} \circ^{n+1} (\mathbf{x}(0), \mathbf{U}) &= \mathbf{f}(\mathbf{f} \circ^n (\mathbf{x}(0), \mathbf{U}), \mathbf{u}(n); \Delta t) \end{aligned}$$

so $\tilde{\mathbf{x}}(k)$ satisfy the dynamic equality constraints of (3.1). The penalty function $\rho_p^2$, for $p \in \mathbb{N}$, is evaluated entry-wise for vector-valued inputs and is given by

$$\rho_p^2(z) = \left( \frac{2z - (z_{\max} + z_{\min})}{z_{\max} - z_{\min}} \right)^{2p} \tag{3.3}$$

which enforces constraints of the form $z \in [z_{\min}, z_{\max}]$ in the limit $p \to \infty$. The penalty function $\rho_p^2$ and its limit are illustrated in Fig. 3.1. In our simulations we found $p = 4$ was sufficient for adequate performance. The diagonal matrix $\mathbf{W}$ is made of tunable weights. This penalty function was chosen for its computational simplicity over the typical choices of barrier functions, like log (which has the advantage of being self-concordant).

In standard SS the states would remain in the NLP only as parameters (*i.e.* (2.2) with the first constraint removed). The resulting NLP would then be solved for $\mathbf{U}^*$ and the states $\mathbf{X}$ would be updated in a separate integration loop. These states would then be fed as parameters back to the NLP and the process would be repeated until some termination

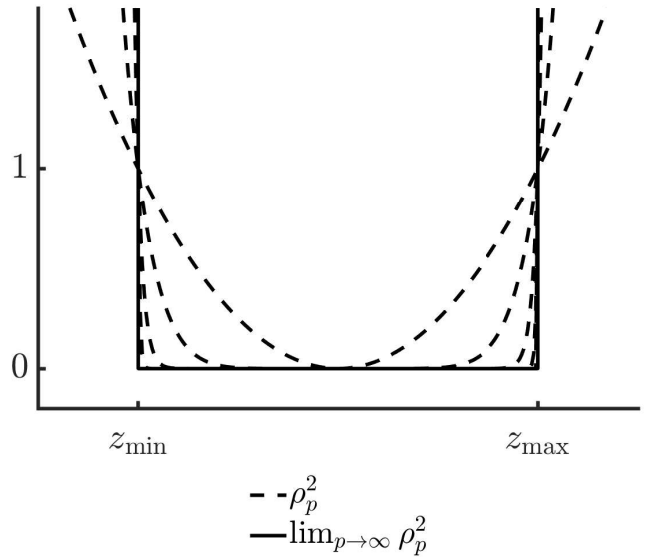



Figure 3.1: The penalty function $\rho_p^2$.

criterion is met. The symSS approach essentially embeds the integration directly in the cost function itself, negating the need for a separate loop. We note that this embedding is only possible provided the plant model $\mathbf{f}$ can be evaluated explicitly. Further, by using symbolic computing combined with automatic differentiation we can compute the exact gradient and Hessian of (3.2) to produce fast solvers and eliminate the need for computationally expensive and possibly unstable numerical differentiation and integration steps.

One key advantage of the symSS formulation (3.2) is the minimal dimension of the resulting NLP as it eliminates the states as degrees of freedom from the original FHOCP (2.2). In direct collocation (DC), multipliers are introduced to handle the dynamic constraints resulting in a larger but sparser NLP. Computational gains can be had if the dense problem of symSS is small enough to be solved faster than the larger sparse problem of DC. Furthermore, symSS avoids the issues that come with initializing physically meaningless multipliers.

A drawback of symSS is its poor scalability. For complex models, the offline computation time of generating the derivatives can be prohibitive *e.g.* for the vehicle model used in this paper (see Appendix A) and for horizons $H > 40$ a typical modern desktop computer will run out of working memory during the derivative generation after more than a day of computing. The computational complexity of the gradient grows linearly and the Hessian quadratically with horizon length $H$. However, as this issue is entirely offline, one can

choose to dedicate more resources to it as desired.

## 3.2 symSS and Code Generation

To efficiently handle an expression like (3.2) and compute its derivatives we first realize it as an ordered sequence of statements whose execution results in (3.2) by introducing intermediate variables. *e.g.* the expression

$$F = D\sin(C\arctan(B\alpha - E(B\alpha - \arctan(B\alpha))))$$

can be realized more efficiently as the sequence

$$\begin{aligned} &t_1 \leftarrow B\alpha, \\ &F \leftarrow -D\sin(C\arctan(-t_1 + E(t_1 - \arctan(t_1)))). \end{aligned}$$

*e.g.* we can realize $f \circ^k (x(0), U)$ as the final value of the following sequence

$$\begin{aligned} &x(1) \leftarrow f(x(0), u(0); \Delta t), \\ &x(2) \leftarrow f(x(1), u(1); \Delta t), \\ &\quad\vdots \\ &x(k) \leftarrow f(x(k-1), u(k-1); \Delta t), \\ &x(k) \end{aligned} \tag{3.4}$$

Note: for complicated functions the statements $x(i+1) \leftarrow f(x(i), u(i); \Delta t)$ may themselves be broken down into ordered subsequences of statements. To compute the derivative of $f \circ^k (x(0), U)$ with respect to an element of $U$ we can call (symbolic) automatic differentiation (AD) routines (*e.g.* `GRADIENT` in Maple) on (3.4) to generate a new sequence of statements whose execution results in the derivative. *e.g.* $\frac{\partial f \circ^k (x(0), U)}{\partial u(0)}$ can be evaluated by executing

the following

$$
\begin{aligned}
x(1) &\leftarrow f(x(0), u(0); \Delta t), \\
df(0) &\leftarrow \frac{\partial f}{\partial u}(x(0), u(0)), \\
x(2) &\leftarrow f(x(1), u(1); \Delta t), \\
df(1) &\leftarrow \frac{\partial f}{\partial x}(x(1), u(1)), \\
&\vdots \\
x(k) &\leftarrow f(x(k-1), u(k-1); \Delta t), \\
df(k-1) &\leftarrow \frac{\partial f}{\partial x}(x(k-1), u(k-1)), \\
df\,du &\leftarrow df(k-1) \times \cdots \times df(1) \times df(0), \\
df\,du &
\end{aligned}
\tag{3.5}
$$

where the statements assigning $df(i)$ maybe be broken down into subsequences. Note: (3.5) is for exposition only. We do not need to derive such a sequence ourselves as the AD routine will do this once we have defined (3.4).

Once we have a sequence to evaluate (3.2) we can generate code to evaluate its exact derivatives by calling AD routines. The resulting code can be further optimized using code optimization tools to yield fast NLP solvers. All of this can be done offline. In this thesis we used Maple which is a software application that allows us to generate the computational sequence symbolically, carry out code optimization, automatic differentiation and generate MATLAB or C code entirely within a single application. The symSS approach is used throughout this thesis and applications of it can be found in Sections 4.2.4, 4.3.3, 5.3.2, 5.3.3, 6.2.1, 7.5.1, 7.5.2, 7.5.3 and 7.5.4.

# Chapter 4

# Accelerating MPCs Using Restricted Lagrangians

In this chapter we present a strategy to reduce the computational burden of MPCs by a novel application of proper orthogonal decomposition (POD). We are able to reduce the dimension of the finite horizon optimal control problem (FHOCP) found within each timestep of an MPC. This is done by applying POD to the Lagrangian of the FHOCP — not the plant model. We extract an affine subspace of reduced dimension that best captures the trajectory of the FHOCP solutions over the controller timesteps. Then we restrict the Lagrangian to that affine subspace. POD shares similarities to principal component analysis and our method can be seen as an application used in machine learning of feature projection to dimensional reduction [28, 29].

## 4.1 POD and Model Reduction

POD is one of the most successful and widely utilized model reduction strategies. It is a reduction method that stems from linear theory but has nonetheless been successfully applied to a wide variety of nonlinear models. Applications include modeling and control of in-cylinder flow [30], the heat diffusion equation [31, 32], fluid channel flow [4], distributed reactor systems [33] and vehicle dynamics [34], to list a few. POD has been applied to control problems by reducing the order of the original plant models. Interestingly, POD when applied to a nonlinear model does not reduce its computational burden for fixed-step integrators typically used in controls applications. However, due to the reduction of model

order achieved by the proposed MPC method, the dimension of the resulting optimization problem is reduced and a speedup in controller computation has been observed [35].

We review the basics of POD here. Suppose we are given a model represented by a system of explicit ordinary differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0$$

where $\mathbf{x} \in \mathbb{R}^n$. The POD-reduced model can be found by the following algorithm:

1. Construct a zero-mean snapshot matrix $\mathbf{X} = \begin{pmatrix} \mathbf{x}(t_1) - \hat{\mathbf{x}} & \cdots & \mathbf{x}(t_N) - \hat{\mathbf{x}} \end{pmatrix} \in \mathbb{R}^{n \times N}$ made up of $N > n$ observations of the model over a collection of simulations where $\hat{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}(t_i)$.

2. Compute the singular value decomposition (SVD) of the snapshot matrix $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ and then truncate $\mathbf{U}$ to its first $r < n$ columns to form $\mathbf{U}_r \in \mathbb{R}^{n \times r}$.

3. Construct the reduced order model using a Galerkin projection

$$\dot{\mathbf{y}}(t) = \mathbf{U}_r^T \mathbf{f}(\mathbf{U}_r \mathbf{y}(t) + \hat{\mathbf{x}}), \quad \mathbf{y}(0) = \mathbf{U}_r^T(\mathbf{x}_0 - \hat{\mathbf{x}}) \tag{4.1}$$

   where $\mathbf{y}(t) \in \mathbb{R}^r$ are coordinates of the reduced space. The original states can be approximated via $\mathbf{x}_r(t) = \mathbf{U}_r \mathbf{y}(t) + \hat{\mathbf{x}} \approx \mathbf{x}(t)$.

Underlying POD is a data compression problem. POD captures the system's output data in as few dimensions as possible by solving the following constrained optimization problem for a fixed $r < n$,

$$\begin{aligned} \min_{\mathbf{U}_r \in \mathbb{R}^{n \times r}} \quad & \sum_{i=1}^{N} \left\| (\mathbf{I} - \mathbf{U}_r \mathbf{U}_r^T)(\mathbf{x}(t_i) - \hat{\mathbf{x}}) \right\|^2 \\ \text{subject to} \quad & \mathbf{U}_r^T \mathbf{U}_r = \mathbf{I} \end{aligned}$$

where $\|\cdot\|$ denotes the Euclidean norm. The truncated $\mathbf{U}_r$ found in step 2) of the algorithm presented above is the optimal solution to this constrained optimization problem [36]. Typically $r$ is chosen such that $\frac{\sum_{k=r+1}^{n} \sigma_k^2}{\sum_{k=1}^{n} \sigma_k^2} < \epsilon$ where $\sigma_k \in \sigma(\mathbf{X})$ and $\sigma(\mathbf{X}) = [\sigma_1, \sigma_2, \ldots, \sigma_n]$ denotes the list of singular values of $\mathbf{X}$ in descending order and $0 < \epsilon < 1$ is a threshold close to 0.

We note that $\mathbf{U}$ is an orthogonal matrix, so that $\mathbf{P}_r = \mathbf{U}_r \mathbf{U}_r^T \in \mathbb{R}^{n \times n}$ is an orthogonal projection matrix onto $\langle \mathbf{U}_r \rangle$, which denotes the column space of $\mathbf{U}_r$. From this we can see that (4.1) is equivalent to

$$\dot{\mathbf{x}}_r(t) = \mathbf{P}_r \mathbf{f}(\mathbf{x}_r(t)), \quad \mathbf{x}_r(0) = \mathbf{P}_r \mathbf{x}_0 + \mathbf{P}_r^{\perp} \hat{\mathbf{x}}$$

which is the projection of the restriction of $\mathbf{f}$ to the affine space $\langle \mathbf{U}_r \rangle + \hat{\mathbf{x}}$ where $\mathbf{P}_r^{\perp} = \mathbf{I} - \mathbf{P}_r$ is the projection matrix orthogonal to $\mathbf{P}_r$ and $\mathbf{x}_r \in \langle \mathbf{U}_r \rangle + \hat{\mathbf{x}}$.

It is important to note that model reduction via POD requires an offline stage where the snapshot matrix is computed using simulations of the original model. Different choices of snapshot matrix will lead to different error statistics of the reduced model, and for nonlinear models, choosing which simulations are best to use in constructing your snapshot matrix remains a difficult problem; see [37].

## 4.2 POD Reduced MPC

Much of the literature directed towards alleviating the bottleneck presented by the hard real-time constraint on MPC has focused on tailoring the online solvers [18, 17, 38]. These often take advantage of the structure of the finite horizon optimal control problem (FHOCP) within MPC, such as condensing methods [39, 40] or structure-exploiting convex solvers [41, 42, 43]. The reduction strategy presented here is novel in that it doesn't target the solver but the problem itself.

Because an MPC solves a sequence of related FHOCPs that are interconnected through the plant dynamics and state feedback, we might expect to be able to extract some features amongst the sequence of optima. In particular, if we can find some affine subspace within which the sequence of optima lies, then restricting the Lagrangian of the FHOCP to this subspace will result in a smaller online optimization problem while introducing minimal error. The following illustrates how we can use the method of POD [44], a method well-known in nonlinear model reduction, to find such a subspace leading to a restricted Lagrangian.

Suppose we have an MPC whose FHOCP has an associated Lagrangian given by $\mathcal{L}$ with optimization variables $\mathbf{z} \in \mathbb{R}^l$ which may include states, controls, multipliers and other variables introduced in formulating the FHOCP. We denote the solution of the FHOCP $\min_{\mathbf{z}} \mathcal{L}(\mathbf{z})$ at timestep $t_i$ by $\mathbf{z}^*(t_i)$.

### 4.2.1 Algorithm

The following steps detail the construction of a restricted Lagrangian using POD for MPC:

1. Construct a snapshot matrix $\mathbf{Z} = \begin{pmatrix} \Delta\mathbf{z}(t_1) & \cdots & \Delta\mathbf{z}(t_N) \end{pmatrix} \in \mathbb{R}^{l \times N}$ where $\Delta\mathbf{z}(t_i) = \mathbf{z}^*(t_i) - \mathbf{z}^*(t_{i-1})$ by running the MPC over representative control scenarios with $N > l$ timesteps.

2. Compute the SVD of the snapshot matrix $\mathbf{Z} = \mathbf{U}\Sigma\mathbf{V}^T$ and then truncate $\mathbf{U}$ to its first $r < l$ columns to form $\mathbf{U}_r \in \mathbb{R}^{l\times r}$.

3. Construct the restricted Lagrangian

$$\overline{\mathcal{L}}(\mathbf{y};\mathbf{z}') = \mathcal{L}(\mathbf{U}_r\mathbf{y} + \mathbf{z}')$$

where $\mathbf{z}' \in \mathbb{R}^l$ and $\mathbf{y} \in \mathbb{R}^r$ are coordinates of the affine space $\langle \mathbf{U}_r \rangle + \mathbf{z}'$ where $\langle \mathbf{U}_r \rangle$ denotes the column space of $\mathbf{U}_r$.

These steps can be conducted entirely offline.

In a POD reduced NMPC (PODrNMPC) we use the restricted Lagrangian constructed above in the online optimization problem. In particular, at timestep $t_i$ we solve $\min_{\mathbf{y}} \overline{\mathcal{L}}(\mathbf{y};\mathbf{z}'(t_i))$ where $\mathbf{z}'(t_i) = \mathbf{z}_r^*(t_{i-1})$ and $\mathbf{z}_r^* = \mathbf{U}_r\mathbf{y}^* + \mathbf{z}' \approx \mathbf{z}^*$ is an approximation of the optimum in the original space. The sequence of solutions of PODrNMPC is given by $\{\mathbf{z}_r^*(t_i)\}_{i=1}^{\infty}$.

The computational savings of PODrNMPC method comes from the dimensional reduction of the restricted Lagrangian. Since the derivatives

$$\nabla\overline{\mathcal{L}}(\mathbf{y};\mathbf{z}') = \mathbf{U}_r^T\nabla\mathcal{L}(\mathbf{U}_r\mathbf{y} + \mathbf{z}') \in \mathbb{R}^r, \tag{4.2}$$

$$\mathbf{H}_{\overline{\mathcal{L}}}(\mathbf{y};\mathbf{z}') = \mathbf{U}_r^T\mathbf{H}_{\mathcal{L}}(\mathbf{U}_r\mathbf{y} + \mathbf{z}')\mathbf{U}_r \in \mathbb{R}^{r\times r} \tag{4.3}$$

are of a smaller dimension, the resulting linear solve within a Newton iteration is more easily computed. The reduced Newton steps $\Delta\mathbf{y}_{NS}^{(k)}$ can be computed by solving the following system of $r$ linear equations

$$\mathbf{U}_r^T\mathbf{H}_{\mathcal{L}}(\mathbf{U}_r\mathbf{y}^{(k)} + \mathbf{z}')\mathbf{U}_r\Delta\mathbf{y}_{NS}^{(k)} = -\mathbf{U}_r^T\nabla\mathcal{L}(\mathbf{U}_r\mathbf{y}^{(k)} + \mathbf{z}') \tag{4.4}$$

Since $\mathbf{U}$ is orthogonal this system is solvable when the original system (2.5) with $\mathbf{z}^{(k)} = \mathbf{U}_r\mathbf{y}^{(k)} + \mathbf{z}'$ is solvable [45]. In the reduced coordinates the initial guess is always $\mathbf{y}^{(0)} = 0$ (except the initial timestep) by construction since the term $\mathbf{z}'$ gets updated between timesteps. At the very first timestep $\mathbf{y}^{(0)}$ and $\mathbf{z}'$ are defined by $\mathbf{z}^{(0)} = \mathbf{U}_r\mathbf{y}^{(0)} + \mathbf{z}'$ where $\mathbf{z}^{(0)}$ is the initial guess. Note that the original gradient and Hessian still need to be computed, but using a symSS approach to define $\mathcal{L}$ (see Section: 3.1) these too can be reduced [46].

From (4.3) we can see that if $\mathbf{H}_{\mathcal{L}}$ is positive definite then so to is $\mathbf{H}_{\overline{\mathcal{L}}}$ by applying Sylvester's criterion and noting that $\mathbf{U}$ is orthogonal. This means if the original Newton step $\Delta\mathbf{z}_{NS}^{(k)}$ can be computed so too can the reduced Newton step $\Delta\mathbf{y}_{NS}^{(k)}$. We also note that $\mathbf{H}_{\overline{\mathcal{L}}}$ inherits the symmetry of $\mathbf{H}_{\mathcal{L}}$ but not its sparsity. Since the reduced problem is

nothing but an unconstrained optimization problem we expect the same rate of quadratic convergence to the reduced optimum $\mathbf{y}^*$.

We call $\overline{\mathcal{L}}$ a restricted Lagrangian since the solution of $\min_{\mathbf{z}} \overline{\mathcal{L}}(\mathbf{z})$ is equal to the solution of the following linearly constrained optimization problem

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathcal{L}(\mathbf{z}) \\ \text{subject to} \quad & (\mathbf{U}_r^{\perp})^T \mathbf{z} = (\mathbf{U}_r^{\perp})^T \mathbf{z}^{(0)} \end{aligned}$$

where $\mathbf{U}_r^{\perp} \in \mathbb{R}^{l \times l-r}$ is the matrix made up of the last $l-r$ columns of $\mathbf{U}$. In this light $\overline{\mathcal{L}}$ is constructed by restricting $\mathcal{L}$ to a linear constraint submanifold and then eliminating those constraints [20]. This approach turns out to be very similar to the idea for mechanical model reduction found in [47]. There they propose a model reduction method for mechanical systems by first imposing constraints to restrict the Lagrangian to a constraint submanifold and then deriving the equations of motion via the Euler-Lagrange equations. In this way the reduced dynamics will still satisfy many truly mechanical properties like preservation of symmetries and energy. As minima correspond to stationary orbits we are trying to accomplish the same task as in [47] but for a different purpose. Further, we have proposed an explicit algorithm to do so using POD.

### 4.2.2 Error Analysis

We now examine the error introduced when using the restricted Lagrangian $\overline{\mathcal{L}}$ instead of $\mathcal{L}$ during optimization. We can maintain good performance because the error incurred is related to the truncated POD modes. We denote the projection matrix onto $\langle \mathbf{U}_r \rangle$ by $\mathbf{P}_r = \mathbf{U}_r \mathbf{U}_r^T \in \mathbb{R}^{l \times l}$ and its orthogonal complement by $\mathbf{P}_r^{\perp} = \mathbf{I} - \mathbf{P}_r$. In Fig. 4.1 we present a depiction of the original and restricted optimization problems.

Recall that $\mathbf{U}_r$ is an optimal solution to

$$\begin{aligned} \min_{\mathbf{U}_r \in \mathbb{R}^{l \times r}} \quad & \left\{ \|\mathbf{P}_r^{\perp} \mathbf{Z}\|_F^2 = \sum_{i=1}^{N} \left\| \mathbf{P}_r^{\perp} \Delta \mathbf{z}(t_i) \right\|^2 \right\} \\ \text{subject to} \quad & \mathbf{U}_r^T \mathbf{U}_r = \mathbf{I} \end{aligned}$$

for fixed $r < l$ where $\|\cdot\|_F$ is the Frobenius norm. Given the optimal $\mathbf{U}_r$ we have the relation

$$\|\mathbf{P}_r^{\perp} \mathbf{Z}\|_F^2 = \sum_{k=r+1}^{l} \sigma_k^2$$

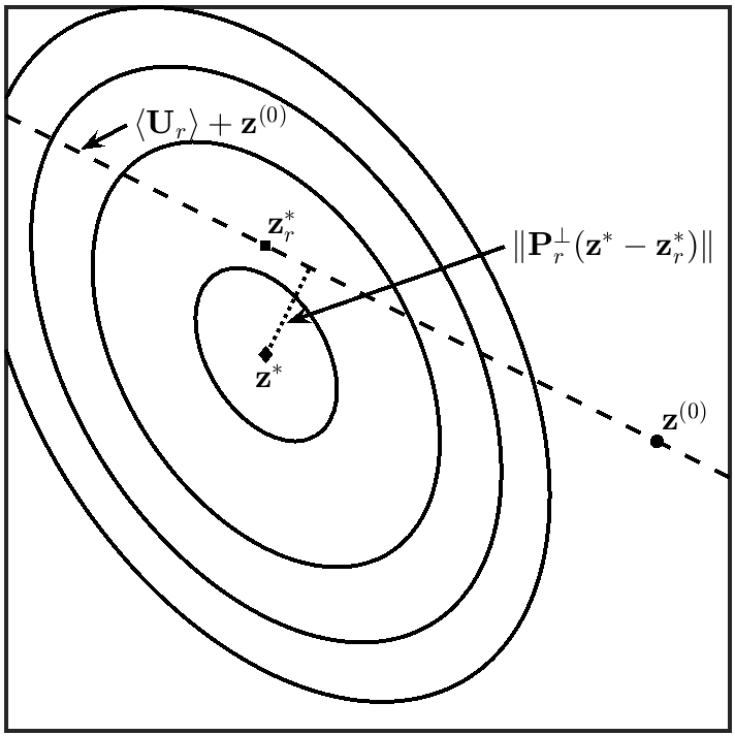



Figure 4.1: A depiction of optimization (Lagrangian represented by the contours) when restricted to an affine subspace. We can see that if $\|\mathbf{P}_r^\perp(\mathbf{z}^* - \mathbf{z}_r^*)\| = 0$ then $\mathbf{z}^* = \mathbf{z}_r^*$ and no error is incurred by approximating the original minimum with that of the restricted minimum.

where $\sigma_k \in \sigma(\mathbf{Z})$ and $\sigma(\mathbf{A})$ denotes the list of singular values of matrix $\mathbf{A}$ in descending order. It has been observed that the POD modes (squares of the singular values of the snapshot matrix) typically shrink, meaning that for even relatively small $r$ the sum of the truncated modes is quite small.

If we assume that the snapshot matrix is generated with $\mathbf{z}^* - \mathbf{z}^{(0)}$ as one of the columns (this assumption is reasonable for MPC since, in practice, we often select the initial guess at a timestep to be the previous timestep's solution, *i.e.* $\mathbf{z}^{(0)}(t_i) = \mathbf{z}^*(t_{i-1})$) then the error between the actual and restricted minimum is

$$\begin{aligned}\|\mathbf{z}^* - \mathbf{z}_r^*\|^2 &= \|\mathbf{P}_r(\mathbf{z}^* - \mathbf{z}_r^*)\|^2 + \|\mathbf{P}_r^\perp(\mathbf{z}^* - \mathbf{z}_r^*)\|^2 \\ &\leq \|\mathbf{P}_r(\mathbf{z}^* - \mathbf{z}_r^*)\|^2 + \sum_{k=r+1}^{l} \sigma_k^2\end{aligned}$$

since $\mathbf{P}_r^{\perp}(\mathbf{z}^* - \mathbf{z}_r^*) = \mathbf{P}_r^{\perp}(\mathbf{z}^* - \mathbf{z}^{(0)})$. The quantity $\|\mathbf{P}_r(\mathbf{z}^* - \mathbf{z}_r^*)\|$ is dependent on $\mathcal{L}$ and the initial guess $\mathbf{z}^{(0)}$ for which no simple bound has been found. In the following we provide a way to estimate this quantity using a quadratic approximation of $\mathcal{L}$ about $\mathbf{z}^*$.

In the case $\mathcal{L} = \frac{1}{2}\mathbf{z}^T\mathbf{Q}\mathbf{z} + \mathbf{c}^T\mathbf{z} + \mathbf{b}$ is a convex quadratic we can compute an exact bound for $\|\mathbf{P}_r(\mathbf{z}^* - \mathbf{z}_r^*)\|$. We know $\mathbf{z}^* = -\mathbf{Q}^{-1}\mathbf{c}$ and given an initial guess $\mathbf{z}^{(0)}$ then $\mathbf{z}_r^* = \mathbf{U}_r\mathbf{y}^* + \mathbf{z}^{(0)}$ where $\mathbf{y}^* = -[\mathbf{U}_r^T\mathbf{Q}\mathbf{U}_r]^{-1}\mathbf{U}_r^T(\mathbf{c} + \mathbf{Q}\mathbf{z}^{(0)})$. Thus

$$\|\mathbf{P}_r(\mathbf{z}^* - \mathbf{z}_r^*)\| = \|\mathbf{U}_r[\mathbf{U}_r^T\mathbf{Q}\mathbf{U}_r]^{-1}\mathbf{U}_r^T(\mathbf{c} + \mathbf{Q}\mathbf{z}^{(0)}) + \mathbf{P}_r(\mathbf{z}^* - \mathbf{z}^{(0)})\| \tag{4.5}$$

and with some manipulation we can find a bound

$$\begin{aligned}
\|\mathbf{P}_r(\mathbf{z}^* - \mathbf{z}_r^*)\| &= \|\mathbf{U}_r(\mathbf{U}_r^T - [\mathbf{U}_r^T\mathbf{Q}\mathbf{U}_r]^{-1}\mathbf{U}_r^T\mathbf{Q})(\mathbf{z}^* - \mathbf{z}^{(0)})\| \\
&\leq \|\mathbf{U}_r[\mathbf{Q}_1^{\mathbf{U}}]^{-1}\mathbf{Q}_2^{\mathbf{U}}(\mathbf{U}_r^{\perp})^T\|\|\mathbf{z}^* - \mathbf{z}^{(0)}\| \\
&\leq \|[\mathbf{Q}_1^{\mathbf{U}}]^{-1}\mathbf{Q}_2^{\mathbf{U}}\|\|\mathbf{z}^* - \mathbf{z}^{(0)}\| \\
&\leq \frac{\max\sigma(\mathbf{Q}_2^{\mathbf{U}})}{\min\sigma(\mathbf{Q}_1^{\mathbf{U}})}\|\mathbf{z}^* - \mathbf{z}^{(0)}\| \\
&\leq \kappa(\mathbf{Q})\|\mathbf{z}^* - \mathbf{z}^{(0)}\|
\end{aligned}$$

where we have written $\mathbf{U}^T\mathbf{Q}\mathbf{U} = \begin{pmatrix} \mathbf{Q}_1^{\mathbf{U}} & \mathbf{Q}_2^{\mathbf{U}} \\ \mathbf{Q}_3^{\mathbf{U}} & \mathbf{Q}_4^{\mathbf{U}} \end{pmatrix}$ in block form with $\mathbf{Q}_1^{\mathbf{U}} \in \mathbb{R}^{r\times r}$ and $\kappa(\mathbf{Q}) = \frac{\max\sigma(\mathbf{Q})}{\min\sigma(\mathbf{Q})}$ is the condition number of $\mathbf{Q}$. The last inequality follows from the Cauchy interlacing theorem and Theorem 1 of [48]. This bound can be used to better estimate the error in the general case by replacing $\mathbf{Q}$ with $\mathbf{H}_{\mathcal{L}}(\mathbf{z}^*)$ to get the following approximate bound

$$\|\mathbf{z}^* - \mathbf{z}_r^*\|^2 \lesssim \kappa(\mathbf{H}_{\mathcal{L}}(\mathbf{z}^*))^2\|\mathbf{z}^* - \mathbf{z}^{(0)}\|^2 + \sum_{k=r+1}^{l}\sigma_k^2.$$

Thus we can say, with some confidence, that provided the Hessian is well-conditioned we can control the error of the restricted Lagrangian solution by selecting $r$ large enough and having a good initial guess $\mathbf{z}^{(0)}$.

The above illustrates how we can expect a restricted Lagrangian to have acceptable error for a particular optimization problem. Since MPC solves a sequence of optimization problems we are, in fact, more interested in the cumulative error when using PODrMPC. We are especially interested in the error incurred from adding the linear constraints. To examine this we compute the error orthogonal to the constraint submanifold.

Using the rules of PODrMPC we can see that

$$\mathbf{z}_r^*(t_j) = \tilde{\mathbf{U}}_r \left( \sum_{i=0}^{j} \mathbf{y}^*(t_i) \right) + \mathbf{z}^{(0)}(t_0) \in \langle \tilde{\mathbf{U}}_r \rangle + \mathbf{z}^{(0)}(t_0)$$

where we have assumed the snapshot $\tilde{\mathbf{Z}}$ with right singular vectors $\tilde{\mathbf{U}}$ is constructed using a simulation with solution sequence $\{\tilde{\mathbf{z}}^*(t_i)\}_{i=0}^N$. This relation tells us the importance of the initial guess $\mathbf{z}^{(0)}(t_0)$ since it will determine the orthogonal component of the error for all PODrMPC solutions. Now it is important to note that an NLP solver will typically require more steps than those of just the PODrMPC we outlined, such as constraint projection, meaning that this relation may be invalidated. Supposing we let $\{\mathbf{z}^*(t_i)\}_{i=0}^N$ be the full MPC solution sequence beginning with initial guess $\mathbf{z}^{(0)}(t_0)$. Then at timestep $t_j$ we have

$$\begin{aligned}
\mathbf{z}^*(t_j) - \mathbf{z}_r^*(t_j) &= \mathbf{z}^*(t_j) + \sum_{k=0}^{s} \tilde{\mathbf{z}}^*(t_k) - \sum_{k=0}^{s} \tilde{\mathbf{z}}^*(t_k) - \tilde{\mathbf{U}}_r \left( \sum_{k=0}^{j} \mathbf{y}^*(t_k) \right) - \mathbf{z}^{(0)}(t_0) \\
&= [\mathbf{z}^*(t_j) - \tilde{\mathbf{z}}^*(t_s)] + \sum_{k=1}^{s} \Delta\tilde{\mathbf{z}}^*(t_k) + [\tilde{\mathbf{z}}^*(t_0) - \mathbf{z}^{(0)}(t_0)] - \tilde{\mathbf{U}}_r \left( \sum_{k=0}^{j} \mathbf{y}^*(t_k) \right)
\end{aligned}$$

From the above we can see

$$\|\tilde{\mathbf{P}}_r^{\perp}[\mathbf{z}^*(t_j) - \mathbf{z}_r^*(t_j)]\| \leq \max_s \|\tilde{\mathbf{P}}_r^{\perp}[\mathbf{z}^*(t_j) - \tilde{\mathbf{z}}^*(t_s)]\| + \|\tilde{\mathbf{P}}_r^{\perp}[\tilde{\mathbf{z}}^*(t_0) - \mathbf{z}^{(0)}(t_0)]\| + \|\tilde{\mathbf{P}}_r^{\perp}\tilde{\mathbf{Z}}\|_F. \quad (4.6)$$

Thus the cumulative orthogonal error of PODrNMPC can be controlled by choosing $\mathbf{z}^{(0)}(t_0) = \tilde{\mathbf{z}}^*(t_0)$ and selecting a snapshot that behaves closely to the actual control scenario. We note that if in fact the snapshot is generated using $\{\mathbf{z}(t_i)\}_{i=0}^N$ (*i.e.* the actual control scenario), the first term can be eliminated entirely.

### 4.2.3 Snapshot Analysis

The snapshot matrix plays an important role in the construction of the reduced Lagrangian and ultimately the performance of PODrMPC. Understanding how to best choose your snapshot matrix is a difficult task but would provide both theoretical and practical benefits to the approach presented here. In the scope of model reduction this remains an open question for POD [37, 44].

Let $\mathbf{Z}$ be a snapshot matrix generated from $\{\mathbf{z}^*(t_i)\}_{i=0}^N$ and $\tilde{\mathbf{Z}} = \mathbf{Z} + \delta\mathbf{Z}$ be a perturbation. We denote the singular values of $\mathbf{Z}$ by $\sigma_i$ and the singular values of $\tilde{\mathbf{Z}}$ by $\tilde{\sigma_i} = \sigma_i + \delta\sigma_i$.

We can relate these singular values to the snapshot perturbation by theorems of Weyl [49] and Mirsky [50] thay tell us

$$|\delta\sigma_i| \leq \|\delta\mathbf{Z}\|_2 \text{ for all } 0 \leq i \leq l \text{ and}$$
$$\sum_{i=1}^{l} \delta\sigma_i^2 \leq \|\delta\mathbf{Z}\|_F^2,$$

respectively. Using these we can relate the orthogonal errors arising from the restriction

$$\begin{aligned}\|\tilde{\mathbf{P}}_r^{\perp}\tilde{\mathbf{Z}}\|_F^2 &\leq \sum_{k=r+1}^{l} \tilde{\sigma}_k^2 \\ &\leq (\|\mathbf{P}_r^{\perp}\mathbf{Z}\|_F + \sqrt{(l-r)}\delta\overline{\sigma}(r))^2\end{aligned}$$

where $\tilde{\mathbf{P}}_r, \tilde{\mathbf{P}}_r^{\perp}$ are the projection matrices resulting from $\tilde{\mathbf{Z}}$ via $\tilde{\mathbf{U}}_r$, $r < l$ and $\delta\overline{\sigma}(r) = \max\{|\delta\sigma_{r+1}|, \ldots, |\delta\sigma_l|\}$. Thus

$$\|\tilde{\mathbf{P}}_r^{\perp}\tilde{\mathbf{Z}}\|_F - \|\mathbf{P}_r^{\perp}\mathbf{Z}\|_F \leq \sqrt{(l-r)}\|\delta\mathbf{Z}\|_2$$

From this bound it is clear to see that for smaller $r$ and larger perturbation the difference in bound grows. Combining this result with (4.6) (assuming $\mathbf{z}^{(0)}(t_0) = \tilde{\mathbf{z}}^*(t_0)$) we have

$$\|\tilde{\mathbf{P}}_r^{\perp}[\mathbf{z}^*(t_j) - \mathbf{z}_r^*(t_j)]\| \leq \|\mathbf{P}_r^{\perp}\mathbf{Z}\|_F + (1 + \sqrt{(l-r)})\|\delta\mathbf{Z}\|_F. \tag{4.7}$$

Thus we can bound the cumulative orthogonal error for a snapshot that doesn't match the scenario exactly in terms of the snapshot and the perturbation from the snapshot. This relation provides a way to gauge how close we require our snapshot to be to the scenario in question in order to get a desired level of performance.

### 4.2.4 Autonomous Vehicle Case Studies

To demonstrate the utility of the proposed MPC reduction methods we apply them to two control problems related to autonomous vehicles. The full plant model of both problems is described in Appendix A.

The first problem is formulated as reference tracking problem in order for a vehicle to carry out a double lane change maneuver. This was the first problem the author tackled using the PODrNMPC method. The problem was formulated using direct collocation and solved via a fixed number of Newton iterations.

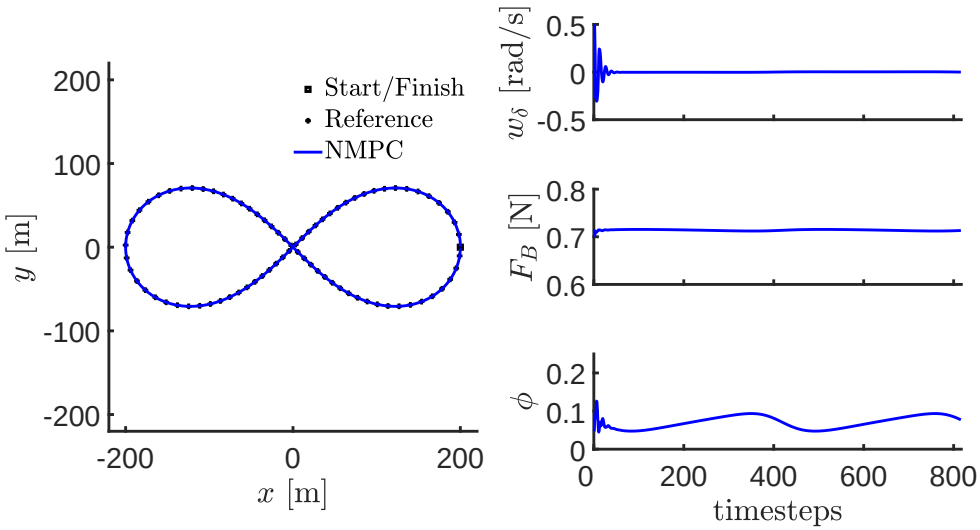



Figure 4.2: The reference NMPC simulation from which the snapshot matrix was generated for the PODrNMPC. On the left is the reference trajectory and controlled vehicle position and on the right is the obtained controls where $w_\delta$ denoted the steering rate, $F_B$ the braking force and $\phi$ the throttle. Note: the reference trajectory excites slow dynamics leading to very smooth controls over the simulation apart from the initial acceleration from rest.

The second problem is a longitudinal tracking/disturbance rejection problem. The goal of the controller is to track a reference velocity while maintaining a straight-line course subject to cross-track disturbance forces. For this problem we utilize a locally linearized plant model in the controller leading to linearized MPC (LMPC) along with the symSS formulation. For this problem the NLP solver consisted of a single Newton step with a constraint projection.

### POD Reduced NMPC for Maneuver Tracking

Control of the car using PODrNMPC was tested using a double lane change maneuver with a constant forward speed of 12 m/s as the reference trajectory. The simulated maneuver is set on a straight road 128 m in length with a lane offset of 3.2 m. The reference trajectory was generated using a piecewise linear interpolation of the maneuver waypoints.

To demonstrate the flexibility of the POD reduction approach, the snapshot matrix for the PODrNMPC was generated using a different reference trajectory, see Fig. 4.2.

The method of direct collocation was used to formulate the FHOCP. A horizon length of $H = 10$ was selected along with a constant timestep of $\Delta t = 100$ ms. The cost function was chosen to minimize controller effort and position tracking error. We included penalty functions in the cost to enforce the inequality constraints on the control inputs. The cost function is given by

$$\mathcal{J}(\mathbf{X}, \mathbf{U}; \mathbf{x}(0)) = \sum_{k=0}^{H-1} \|\mathbf{x}(k) - \mathbf{x}^r(k)\|_{\mathbf{Q}}^2 + \|\mathbf{u}(k)\|_{\mathbf{R}}^2 + \|\varrho_p(\mathbf{h}(\mathbf{U}, \mathbf{X}; \mathbf{x}(0)))\|_1$$

where $\mathbf{Q} = \text{diag}(100, 100, 0, 0, 0, 0, 0)$, $\mathbf{R} = \text{diag}(10, 0.001, 10)$, $\mathbf{h}(\mathbf{U}, \mathbf{X}; \mathbf{x}(0)) \leq 0$ is the vector of inequality constraints and the penalty function, which is evaluated entry-wise for vector-valued arguments, $\varrho_p$, for $p \in \mathbb{N}$, is given by

$$\varrho_p(z) = \begin{cases} 0 & z \leq -1 \\ (z+1)^p & z > -1 \end{cases}.$$

This penalty function was chosen to enforce constraints of the form $z \leq 0$ due to its behaviour in the limit $p \to \infty$. In this study we fixed $p = 8$.

The dynamics of the model are discretized using the explicit Euler method so that at each timestep we minimize $\mathcal{J}$ subject to the following vector of equality constraints

$$\mathbf{g}(\mathbf{X}, \mathbf{U}; \mathbf{x}(0)) = \begin{pmatrix} \mathbf{x}(1) - \mathbf{x}(0) - \Phi(\mathbf{x}(0), \mathbf{u}(0))\Delta t \\ \vdots \\ \mathbf{x}(H) - \mathbf{x}(H-1) - \Phi(\mathbf{x}(H-1), \mathbf{u}(H-1))\Delta t \end{pmatrix}$$

where $\mathbf{x}(0)$ is the initial condition of the FHOCP which is derived from the solution of the previous timestep. The Lagrangian of the FHOCP is then

$$\mathcal{L}(\mathbf{X}, \mathbf{U}, \boldsymbol{\lambda}; \mathbf{x}(0)) = \mathcal{J}(\mathbf{X}, \mathbf{U}; \mathbf{x}(0)) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{X}, \mathbf{U}; \mathbf{x}(0))$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. The dimension of the full FHOCP for this problem is $H(m + 2n) = 170$.

All simulations were run using MATLAB 2016a on a desktop PC with an Intel i7-4790 CPU. Simulations were begun with a warm start computed using `fmincon` to find the initial values of $\mathbf{X}$ and $\mathbf{U}$. The initial Lagrange multipliers were then computed using least-squares [51]. The gradient $\nabla\mathcal{L}$ and Hessian $\mathbf{H}_{\mathcal{L}}$ were computed symbolically using MAPLE-generated optimized code which was then converted to MEX files. Our optimization method employed full Newton steps with only a single iteration per timestep. All

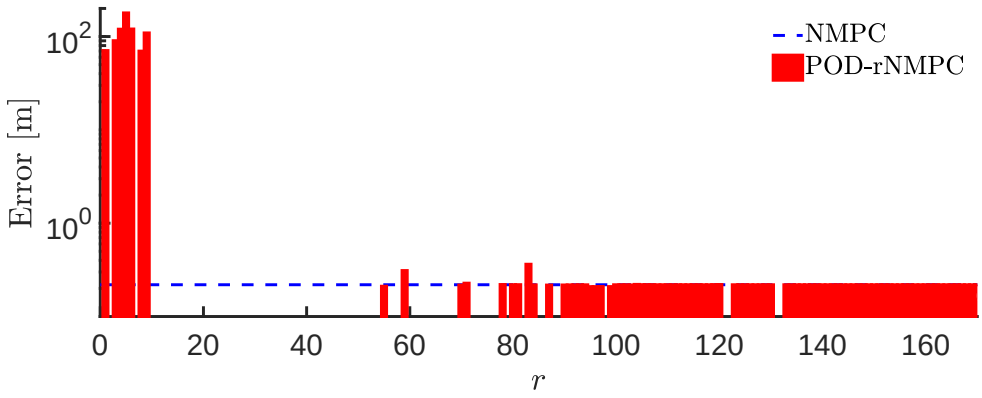



Figure 4.3: The maximum position error of a PODrNMPC over the double lane change maneuver with $r$ variables. We provide the maximum error of the original NMPC as a reference. No bar indicates that the simulation with that number of reduced variables failed to complete.

matrices were declared `sparse` where appropriate to take advantage of MATLAB's sparse solvers and timing data was found using `tic` and `toc`. In our implementation of PODrNMPC the full gradient and Hessian are computed and then transformed within each timestep following (4.4).

By applying the snapshot matrix generated from the figure-eight trajectory to the double lane change maneuver we found that the optimal number of reduced variables was $r = 55$. In Fig. 4.3 we can see the effect that choice of $r$ has on the quality of the PODrNMPC's performance. We note that for many choices of $r$ the PODrNMPC failed to operate over the entirety of the simulation due to the transformed Hessian becoming ill-conditioned at some timestep. In the subsequent discussion we focus on the results of the PODrNMPC with $r = 55$.

The PODrNMPC resulted in turnaround times on average 2 times faster than the original NMPC. In Table 4.1 we present measured computation times averaged over all timesteps over 100 simulations. Despite the fact that the reduced linear system is dense and the original system sparse, the dimensional reduction resulted in a 5.8 times faster solution of the linear system by going from 170 to 55 equations.

As can be seen in Figs. 4.4 and 4.5 the performance of the PODrNMPC is remarkably similar to the original NMPC. Surprisingly, in terms of maximal position error and lateral position error the PODrNMPC marginally improved the performance of the controller re-

Table 4.1: Mean Computation Time of 100 Simulations

| | Linear System Solution Time | Turnaround Time |
|---|---|---|
| NMPC | 0.48 ms | 0.59 ms |
| PODrNMPC (r=55) | 0.083 ms | 0.30 ms |
| PODrNMPC (r=80) | 0.14 ms | 0.39 ms |
| PODrNMPC (r=115) | 0.25 ms | 0.57 ms |

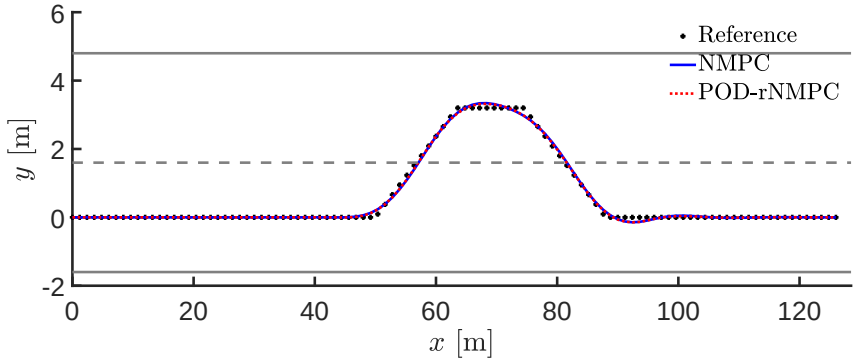



Figure 4.4: Vehicle position during the controlled double lane change maneuver. Note that the axes are not scaled equally. The waypoints defining the maneuver are: $\{(0,0),(0,50),(3.2,63.5),(3.2,74.5),(0,88),(0,128)\}$.

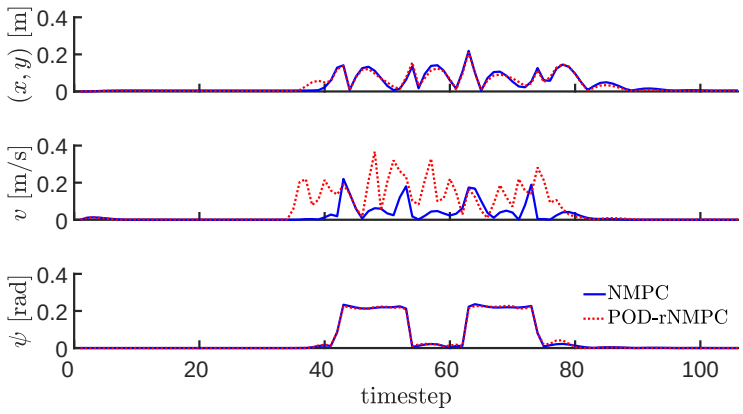



Figure 4.5: The absolute error of position $(x, y)$, speed $(v)$ and yaw $(\psi)$ relative to the reference trajectory over the double lane change maneuver.

ducing the error from 22 cm to 21 cm and from 6.8% to 6.6% of the lane offset, respectively. Greatest error was seen in the speed where the PODrNMPC deviated by at most 3.0% from the reference values compared to the NMPC which deviated at most 1.8%. It is expected this could be improved by including a speed tracking term in the cost function. In Fig. 4.6 we can see that the control inputs of the PODrNMPC are of the same magnitude as the controls of the original NMPC.

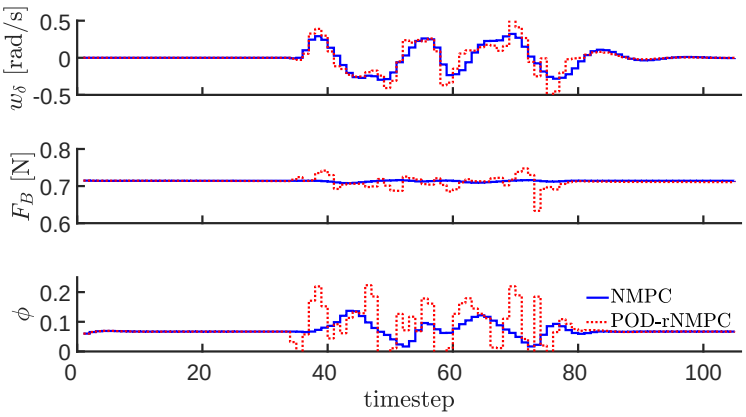



Figure 4.6: The obtained control inputs over the double lane change maneuver.

We do notice that the controls obtained by the PODrNMPC exhibit greater rates of change than the NMPC controls. This is a direct result of the dimensional reduction. On average the PODrNMPC and NMPC inputs for steering angle speed, braking force and throttle position differed by $4.6\%, 4.3 \times 10^{-7}\%$ and $3.0\%$ of the input range, respectively.

We will see in the next sections that we can get past the ill-conditioning causing the ragged appearance of Fig. 4.3 using the symSS approach.

### POD Reduced LMPC for Disturbance Rejection

The example LMPC problem we consider is a longitudinal vehicle controller with disturbance rejection. At a particular timestep we linearize the nominal model about $\bar{\mathbf{x}}_0 = [0, 0, v_0, 0, 0, 0, 0]^T, \bar{\mathbf{u}}_0 = [0, 0, \phi_0]^T$ where $v_0$ is the velocity of the latest measured state and $\phi_0$ is the throttle input of the latest timestep. We used the forward Euler method with the linear plant model to derive a linear discrete time model

$$\mathbf{x}(k+1) = [\mathbf{I} + \Delta t \mathbf{A}]\mathbf{x}(k) + \Delta t \mathbf{B}\mathbf{u}(k) \tag{4.8}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the Jacobians of the nonlinear model with respect to states and controls, respectively, evaluated at $(\bar{\mathbf{x}}_0, \bar{\mathbf{u}}_0)$. More details of the linearization can be found in Appendix A.

The selected cost function is

$$\mathcal{J}(\mathbf{U}; \mathbf{x}(0)) = \sum_{k=1}^{H} \|\tilde{\mathbf{x}}(k) - \mathbf{x}^r(k)\|_{\mathbf{Q}}^2 + \sum_{k=0}^{H-1} \|\mathbf{u}(k)\|_{\mathbf{R}}^2 + \|\Delta\mathbf{u}(k)\|_{\Delta\mathbf{R}}^2 + \|\rho_p(\mathbf{u}(k))\|_{\mathbf{P}}^2 \tag{4.9}$$

where

$$\tilde{\mathbf{x}}(k) = \left( [\mathbf{I} + \Delta t \mathbf{A}]^k \mathbf{x}(0) + \sum_{j=0}^{k-1} [\mathbf{I} + \Delta t \mathbf{A}]^{k-1-j} \Delta t \mathbf{B}\mathbf{u}(j) \right),$$

$\mathbf{x}^r(k)$ is a reference trajectory, $\Delta\mathbf{u}(k) = (\mathbf{u}(k) - \mathbf{u}(k-1))/\Delta t$ with $\Delta\mathbf{u}(0) = 0$, and $\mathbf{Q}, \mathbf{R}, \Delta\mathbf{R}, \mathbf{P}$ are tunable diagonal weighting matrices. Note $\|\mathbf{z}\|_{\mathbf{A}}^2 = \mathbf{z}^T\mathbf{A}\mathbf{z}$ and the half-penalty function $\rho_p$ is evaluated entry-wise for vector-valued arguments. In our example control problem the reference trajectory is given by vectors of the form $\mathbf{x}^r = [0, y^r, v^r, 0, 0, 0, 0]^T$ since we assume the longitudinal motion is along the $x$ axis. This choice of cost function balances two objectives: to minimize the deviation from the reference trajectory and to minimize the inputs and their rates of action.

In our simulations $\Delta t = 50$ms, $H = 9$, $p = 4$, $\mathbf{Q} = \text{diag}(10^4, 10^4, 2 \times 10^4, 0, 100, 0, 0)$, $\mathbf{R} = \text{diag}(500, 100, 200)$, $\Delta\mathbf{R} = \text{diag}(1000, 500, 500)$, $\mathbf{P} = \text{diag}(500, 1, 100)$. The value of $H$

was selected as the minimal value of $H$ that yields satisfactory results given the timestep and other choice of parameters which were found by manual tuning.

For this problem we utilized the symSS formulation implemented in Maple. The symSS Lagrangian and its derivatives were generated and optimized as standalone functions. These functions were then converted to MEX functions that we then used within Newton's method. All simulations were run using MATLAB 2017b on a desktop with an Intel(R) Core(TM) i7-4790 CPU.

We drew our snapshot matrix from a 100s simulation with a constant reference velocity of 10m/s and non-zero disturbance $D_y \sim 10^3 \times \mathcal{N}$, where $\mathcal{N}$ denotes the standard normal distribution. The snapshot of the controls, generated from simulation, is displayed in Fig. 4.7.

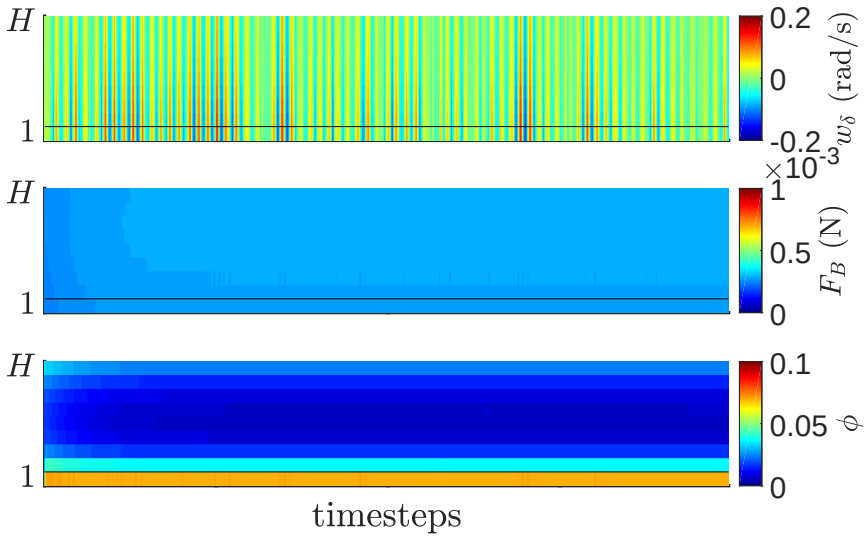



Figure 4.7: Snapshot of the controls over the prediction horizon used to generate a PO-DrLMPC.

In Fig. 4.7 we separate the snapshot matrix into each control inputs' component for display purposes as the inputs vary dramatically in scale. The prediction horizon is displayed on the $y$ axis with the first element in the control horizon at the bottom. Given the constant reference trajectory, we see that brake and throttle inputs are essentially constant over the entire simulation. Further, the brake force is essentially zero given that its maximal value is 6 orders of magnitude greater than what we observe here. The steering rate input is most active since it is constantly acting to counteract the disturbance input.

To see the components underlying the construction of a POD restricted Lagrangian in Fig. 4.8 we display the left singular vectors resulting from the decomposition of the snapshot matrix along with the associated projection matrix for $r = 2$ and the truncation errors for all $r$.

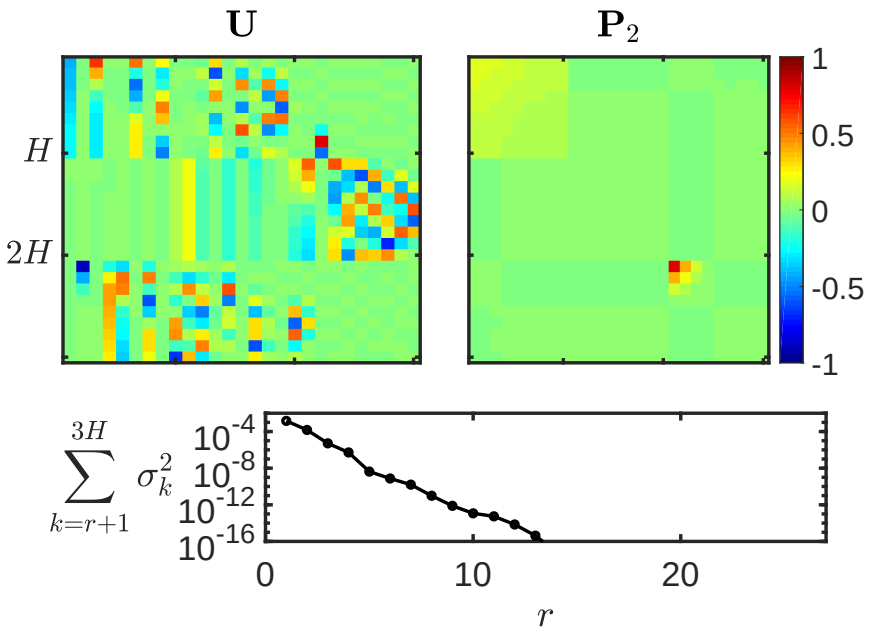



Figure 4.8: Resultant singular vectors, projection matrix and truncation errors of the snapshot matrix.

From these we can see that the states of most importance are the steering rate over most of the horizon and the first and second input of the throttle, by looking at the first 2 or 3 columns of $\mathbf{U}$. The brakes appear entirely unimportant. In this example for $r = 2$, $\sum_{k=r+1}^{3H} \sigma_k^2 = 1.48 \times 10^{-4}$. We can see that the truncation error is quite small for even small values of $r$.

In Fig. 4.9 we display a new simulation that we use to test the PODrLMPC constructed from the snapshot information above. In this scenario the reference velocity is non-constant and the disturbance is only periodically sampled $D_y(t_i) \sim 10^4 \times \delta_{0,i \mod 100} \times |\mathcal{N}|$ where $\delta_{i,j}$ is the Dirac delta. This is to simulate periodic gusts of a cross-wind in a consistent direction.

We display the results of the PODrLMPC with $r = 2$ as compared to an unreduced LMPC in Figs. 4.10 and 4.11.

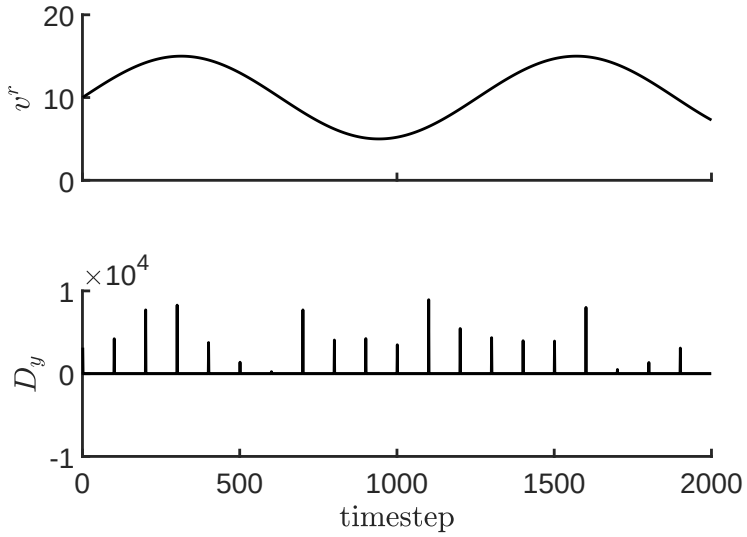



Figure 4.9: Sample PODrLMPC test scenario.

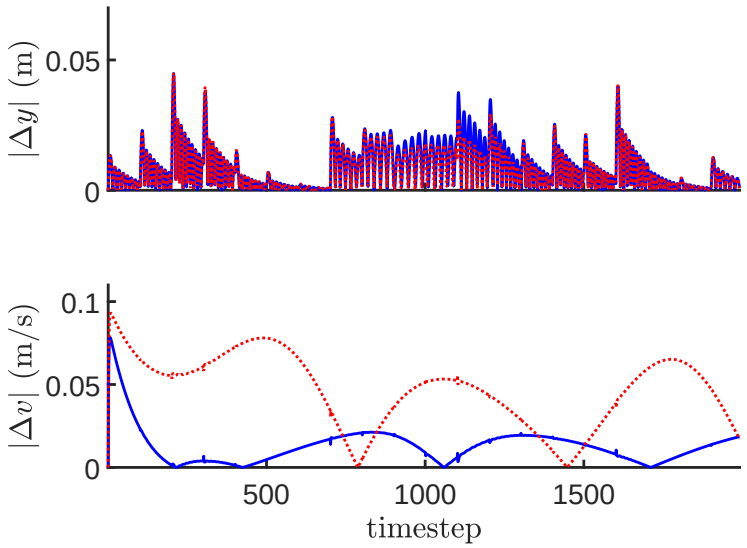



Figure 4.11: Simulation tracking error. LMPC is given in blue (solid), PODrLMPC for $r = 2$ is given in red (dash).

From the first plot in Fig. 4.11 we can see that the PODrLMPC performs as well as the standard LMPC in terms of rejecting the disturbances. It however performs worse in terms of velocity tracking. Given the choice of snapshot, this is expected since the throttle and

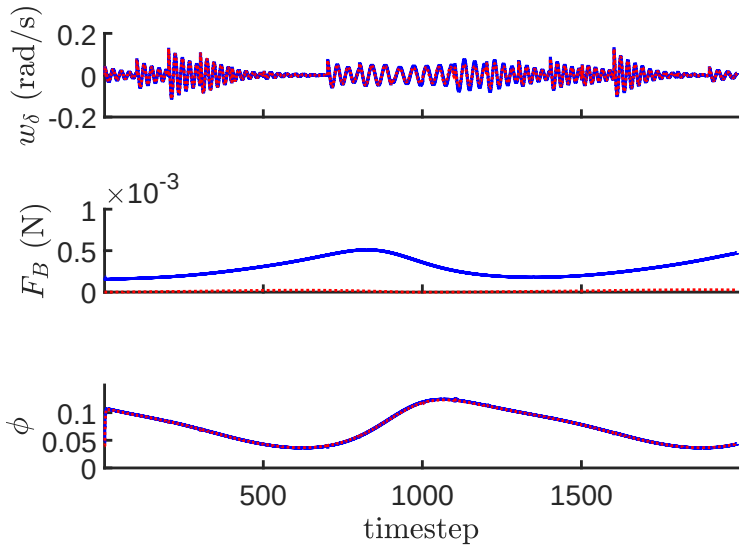



Figure 4.10: Simulation control inputs. LMPC is given in blue (solid), PODrLMPC for $r = 2$ is given in red (dash).

brake are not excited during the snapshot simulation and hence their characteristics are not captured by the subspace of the restricted Lagrangian. This points to the importance of choosing our snapshots wisely in order to capture as much important behaviour as possible. However, with that said the velocity tracking error of the PODrLMPC is at most 1.02% relative to the reference trajectory, compared to 0.77% for the LMPC which is only a 0.25% degradation.

Lastly, in Fig. 4.12 we display the TAT results of the simulation. Comparing the mean TATs, PODrLMPC for $r = 2$ is 1.32× faster than LMPC and with a symbolic solver it is 2.38× faster. Inclusion of the symbolic linear solver helps us take advantage of the dimensional reduction by improving the PODrLMPC TATs by 1.80×, a significant improvement.

To understand the role of the reduction dimension $r$ we display the maximum tracking errors and mean TAT acceleration factors for the selected choice of simulation in Figs. 4.13 and 4.14.

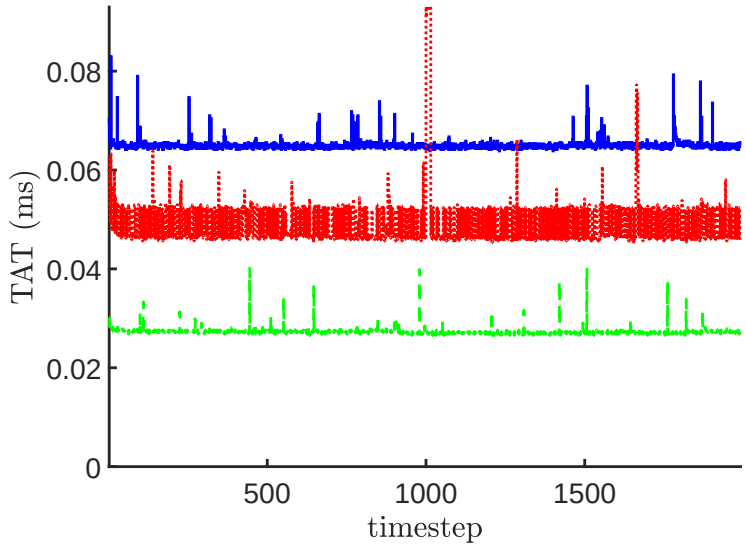



Figure 4.12: Sample TATs of the selected simulation. LMPC is given in blue (solid), PODrLMPC for $r = 2$ is given in red (dash), and PODrLMPC for $r = 2$ with a symbolic linear solver is given in green (long dash).

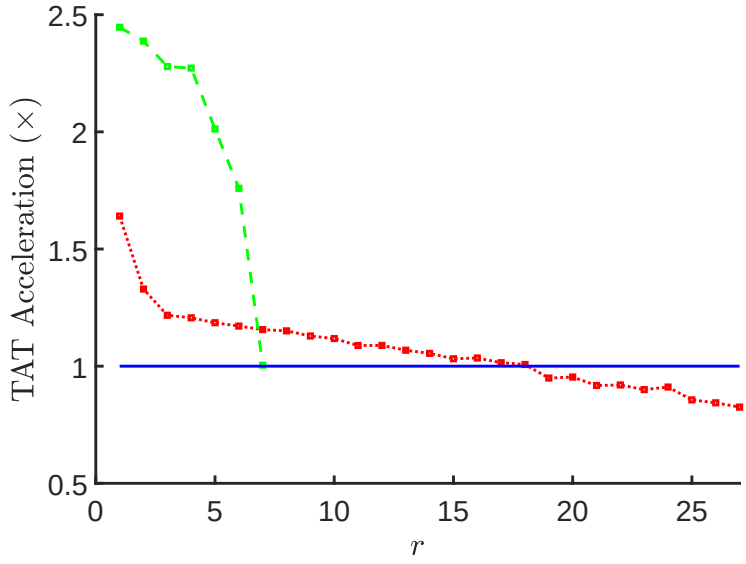



Figure 4.14: Role of $r$ in PODrLMPC TAT acceleration. The acceleration factor of PODrLMPC is given in red (dash) and the acceleration factor using a symbolic linear solver is given in green (long dash). The standard LPMC is given by blue (solid) line, as reference. These results are the average of the mean TAT of 100 repeated simulations. The blue line represents 0.0649ms

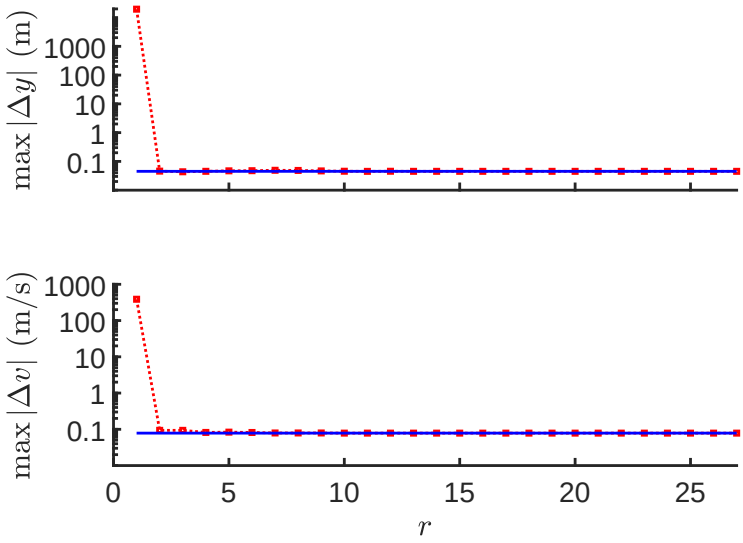



Figure 4.13: Role of $r$ in PODrLMPC tracking error. The error for PODrLMPC is given in red (dash), and the error of LMPC is given by the blue (solid) line as a reference.

We can see that for $r \geq 1$ the PODrLMPC performs very similarly to LMPC in terms of maximal tracking error. Only for $r = 1$ does this error explode. There is some irregularity about small $r$s, *i.e.* $r < H$, where the relation between error and $r$ is not strictly monotonic. The fact that the error plateaus for $r \geq 2$ indicates that the first 2 singular vectors contain almost all the necessary controller information. In terms of mean TAT the acceleration appears to decrease almost linearly with $r$. For $r > 15 \approx mH/2$ there is no computational gain to be had at all using PODrLMPC. When using a symbolic linear solver gains are only to be had for small $r \leq 6$ since the symbolic complexity of the linear solve grows rapidly for larger $r$.

Understanding the role of a snapshot in the success of a PODrMPC is of significant importance to this method. We attempt here to shed some light on this issue. In Fig. 4.15 we display the role of a snapshot on PODrLMPC performance in terms of tracking error in comparison to some other snapshot measures. We generated a variety of snapshots using the constant reference speed of 10m/s and $D_y(t_i) \sim A \times \delta_{0,i \mod 100}$ where $A$ is the snapshot amplitude. The resulting PODrLMPC were then run on the sample trajectory of Fig. 4.9 and their maximal cross-tracking errors for different $r$ are displayed in the last plot of Fig. 4.15. It is evident that choosing disturbance inputs of the snapshot matrix closer to the sample scenario leads to better PODrLMPC performance, *i.e.* a greater reduction is possible. With smaller disturbances, the steering input (which is important for disturbance

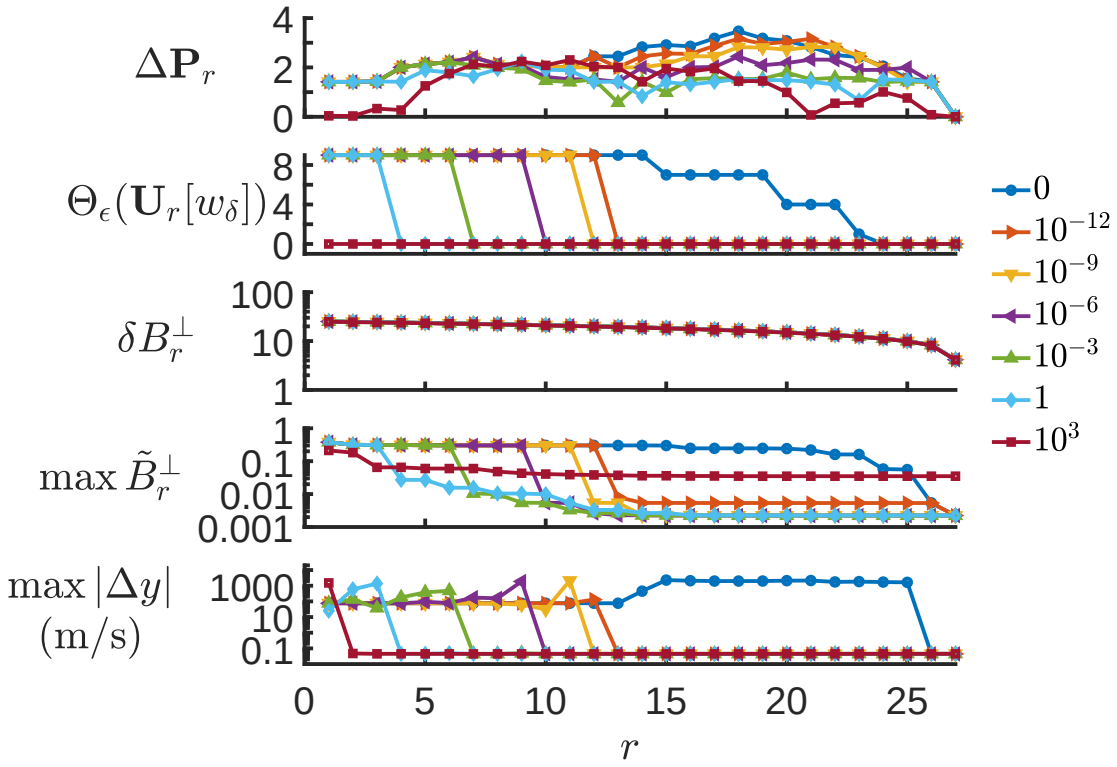



Figure 4.15: The legend displays the amplitude of the disturbances $D_y$ associated to the snapshots of the different PODrLMPCs.

rejection) is not excited meaning the resultant restricted Lagrangian captures less of this important input.

In the first four plots of Fig. 4.15 we present different measures of the snapshot. What we are interested in finding is a function of the snapshot that correlates with the performance of the PODrLMPC corresponding to that snapshot. One point of particular interest is determining the level of reduced dimension $r$ possible from snapshot data alone. The first plot is a standard measure of matrix subspace similarity given by the norm of the difference of the projection matrices onto those subspaces. For this plot $\Delta \mathbf{P}_r = \|\mathbf{P}_r - \mathbf{P}_r^*\|_F$ where $\mathbf{P}_r$ is the projection matrix onto the subspace of the snapshot data and $\mathbf{P}_r^*$ is the projection matrix corresponding to the snapshot given from the example trajectory itself. We can see that this measure does not correlate at all with the last plot and can be ruled out as a candidate measure meaning that selecting snapshots such that the reduced spaces $\mathbf{U}_r$ are 'similar' to $\mathbf{U}_r^*$ is in fact not a good strategy.

In the second plot the measure is given by

$$\Theta_\epsilon(\mathbf{U}_r) = \sum_{i=1}^{mH} \theta(\epsilon - \|\mathbf{U}_r \mathbf{e}_i\|_\infty)$$

where $\theta$ is the Heaviside function, $\epsilon > 0$ is some chosen tolerance, and $\mathbf{e}_i \in \mathbb{R}^{mH}$ is the $i^{\text{th}}$ standard basis vector. This measure counts the number of rows of $\mathbf{U}_r$ that has at least one non-trivial entry where the triviality is determined by $\epsilon$. In our case we selected $\epsilon = \Delta t$ and applied this measure to the submatrix containing only those rows corresponding to the first input $w_\delta$. This measure can be seen to correlate much better with the last plot indicating the importance of 'active' steering input timesteps. This measure is independent of data from the example scenario itself but gave such clear results only when restricted to steering input that we determined to be the most important input for cross-track error rejection. We desire a measure that is independent of such an engineer's insight; however this result provides justification for the method introduced in the next chapter. The sensitivity to 'trivial' rows of $\mathbf{U}_r$ motivates the development of our informed move blocking scheme discussed in Section 4.3.

In the third plot $\delta B_r^\perp$ represents the bound given by the right hand side of (4.7) which is a function of the snapshot from the example trajectory itself and the difference in snapshots $\delta\mathbf{Z}$ from the example trajectory and the one chosen. We can see that for different snapshots the term $\|\delta\mathbf{Z}\|_F$ is not sensitive enough to distinguish them let alone provide the information we desire.

Lastly, in the fourth plot $\tilde{B}_r^\perp$ represents the bound given by the right hand side of (4.6) where max is taken over all timesteps. This is a much tighter bound than that of (4.7) and correlates well with the controller performance. We can see that when $\max \tilde{B}_r^\perp \lesssim 0.1$ the corresponding PODrLMPC has good tracking performance. This is a promising measure since it only requires snapshot data from the sample and example scenario and does not require simulation of the PODrLMPC. Further work is required to refine this result, such as selecting the threshold bound, but our investigations have provided a promising candidate.

## 4.3 An Informed Move Blocking Strategy for MPC

In this section we extend the insights of the PODrMPC method to generate an informed move blocking (IMB) strategy. This method is novel in that it allows different controls to be blocked differently for multi-input systems. Like PODrNMPC this method is best suited to the problem of reducing the turnaround time of a pre-existing controller as it assumes in the first stage that a functioning MPC is already present. In the first stage we apply the algorithm in Section 4.2 to construct a restricted Lagrangian. Once we find the smallest dimension $r$ for which the PODrMPC delivers satisfactory results we use the diagonals of the projection matrix $\mathbf{P}_r$ to inform the construction of a move blocking strategy. By applying MB we can inject sparsity into the reduced problem leading to a greater acceleration of controller turnaround time.

Move blocking (MB) is a strategy developed to reduce the computational complexity of a MPC controller. MB fixes or 'blocks' the control input (or rate of change of control input) to be constant over some timesteps in the horizon. By doing so it reduces the degrees of freedom of the controller and thus simplifies the FHOCP. Thus far MB has been successfully employed in a wide range of MPC problems, *e.g.* a voltage-mode controller for a dc-dc boost converter [52], a diesel engine airpath controller [53] and a building cooling system controller [54]. The theoretical aspects of MB has been investigated primarily for linear systems where it has been found that, to retain recursive feasibility, the blocking strategy must be time-varying [55] or additional constraints need to be added [56, 57]. Optimal and robust MB strategies for linear systems have been presented in [58, 59]. Our IMB strategy is applicable to nonlinear control problems and is more flexible than traditional strategies since different inputs can have different blocking strategies.

### 4.3.1 Move Blocking Basics

Consider a single input system controlled by an MPC with horizon $H$. Let us arrange the controls over the horizon as a single column vector $U = \begin{pmatrix} u(0) & \cdots & u(H-1) \end{pmatrix}^T \in \mathbb{R}^H$. Move blocking reduces the number of degrees of freedom of the FHOCP meaning we only need to solve for an optimal control $U' \in \mathbb{R}^b$ with $b < H$ where $\mathbf{B}U' = U$ and $\mathbf{B} \in \mathbb{R}^{H \times b}$ is a blocking matrix. As an example, if $H = 4$ and we wish to block the second and third

controls then we may have

$$\begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u'_1 \\ u'_2 \\ u'_3 \end{pmatrix}. \tag{4.10}$$

In general, blocking matrices are of the form

$$\mathbf{B} = \bigoplus_{i=1}^{b} \mathbf{1}_{h_i} \in \mathbb{R}^{H \times b}$$

where $\mathbf{1}_{h_i} \in \mathbb{R}^{h_i}$ is a column vector of all ones, $\sum_{i=1}^{b} h_i = H$ and $\oplus$ denotes the direct sum of matrices. This relation means we can identify a blocking matrix using the shorthand $\mathbf{B} = [h_1, h_2, \ldots, h_b]$, so *e.g.* the blocking matrix in (4.10) can be identified with $[1, 2, 1]$.

Blocking matrices (that aren't the identity) can be generated by the appropriate right matrix multiplication of elementary blocking matrices

$$\mathbf{E}(k; l) = \mathbf{I}_{l-1} \oplus \mathbf{1}_2 \oplus \mathbf{I}_{k-1-l} \in \mathbb{R}^{k \times k-1},$$

$1 \leq l \leq k-1, k \geq 2$. An elementary matrix $\mathbf{E}(k; l)$ blocks controls $l$ and $l+1$ and preserves the $H-k$ blocking moves already implemented, *e.g.* the blocking matrix in (4.10) is $\mathbf{E}(4; 2)$. If we then wanted to block the first and second inputs we would use the blocking matrix $\mathbf{B} = \mathbf{E}(4; 2)\mathbf{E}(3; 1)$ so that

$$\begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u'_1 \\ u'_2 \end{pmatrix}.$$

From this it is clear to see that a blocking matrix consisting of $q \geq 1$ blocking moves can be written as

$$\mathbf{B} = \mathbf{E}(H; l_0)\mathbf{E}(H-1; l_1) \cdots \mathbf{E}(H-(q-1); l_{q-1})$$

where $1 \leq l_i \leq H - 1 - i$. This decomposition is not unique, *e.g.* $\mathbf{E}(4; 2)\mathbf{E}(3; 1) = \mathbf{E}(4; 1)\mathbf{E}(3; 1)$ and $\mathbf{1}_H$ is the result of any valid set of $H-1$ blocking moves.

Now let's consider the more general case of a multi-input system, $\mathbf{u} \in \mathbb{R}^m$, and suppose the controls over the horizon are arranged as a single column vector in the following manner $\mathbf{U} = \begin{pmatrix} u_1(0) & \cdots & u_1(H-1) & \cdots & u_m(0) & \cdots & u_m(H-1) \end{pmatrix}^T \in \mathbb{R}^{mH}$. In standard move

blocking schemes the move blocking matrix is given by $\mathbf{M} = \mathbf{I}_m \otimes \mathbf{B} = \bigoplus_{k=1}^{m} \mathbf{B}$ so that $\mathbf{U} = \mathbf{M}\mathbf{U}'$ and all inputs are blocked in the same manner by a blocking matrix $\mathbf{B}$. We generalize this approach here and allow inputs to be blocked via differing strategies. Thus we allow move blocking matrices to take the more general form

$$\mathbf{M} = \bigoplus_{k=1}^{m} \mathbf{B}_k \in \mathbb{R}^{mH \times b}$$

where $\mathbf{B}_k \in \mathbb{R}^{H \times b_k}$ is a blocking matrix, $\sum_{k=1}^{m} b_k = b$ and the reduced controls $\mathbf{U}' \in \mathbb{R}^b$ with $b < mH$. To illustrate, consider a double input system with $H = 4$ and a strategy where we block the second and third controls of the first input and the first, second and third controls of the second input. Then we have

$$\begin{pmatrix} u_1(0) \\ u_1(1) \\ u_1(2) \\ u_1(3) \\ u_2(0) \\ u_2(1) \\ u_2(2) \\ u_2(3) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & & \\ 0 & 1 & 0 & & \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \\ & & & 1 & 0 \\ & & & 1 & 0 \\ & & & 1 & 0 \\ & & & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1' \\ u_2' \\ u_3' \\ u_4' \\ u_5' \end{pmatrix}. \tag{4.11}$$

This strategy may allow for greater dimensional reduction as the dynamics of the different control inputs can be accommodated separately. We use the results of PODrMPC to guide this approach. Note that the same shorthand of blocking matrices can be applied to move blocking matrices, *e.g.* the move blocking matrix in (4.11) can be identified with $[1, 2, 1, 3, 1]$.

Move blocking matrices reduce the cost of computing Newton steps in the same way that the restricted Lagrangians of Section 4.2 do. In fact, move blocking matrices simply generate a restricted Lagrangian with a different constraint manifold. We refer to an MPC that utilizes a move blocking strategy as move blocking reduced MPC (MBrMPC).

Assuming that the Lagrangian is generated using the symSS formulation (so only the controls are optimization variables) then for a move blocking matrix $\mathbf{M} \in \mathbb{R}^{mH \times b}$ we can construct the restricted Lagrangian

$$\mathcal{L}_{MB}(\mathbf{U}') = \mathcal{L}(\mathbf{M}\mathbf{U}')$$

where $\mathcal{L}$ is the original Lagrangian. The reduced Newton step $\Delta\mathbf{U}'^{(k)}_{NS}$ can be computed by solving the following system of $b$ linear equations

$$\mathbf{M}^T\mathbf{H}_{\mathcal{L}}(\mathbf{M}\mathbf{U}'^{(k)})\mathbf{M}\Delta\mathbf{U}'^{(k)}_{NS} = -\mathbf{M}^T\nabla\mathcal{L}(\mathbf{M}\mathbf{U}'^{(k)}). \tag{4.12}$$

Compared to (4.4) this reduced Newton step is more sparse, which can be used to reduce the linear solve computational complexity and thus improve controller turnaround times.

### 4.3.2 Informed Move Blocking Generation

We now illustrate how to compute a sequence of increasingly blocked move blocking matrices using data from a PODrMPC. This method is summarized in Algorithm 1. First, we extract the diagonals of the projection matrix $\mathbf{P}_r = \mathbf{U}_r\mathbf{U}_r^T$ as the vector $\mathbf{p}$. We interpret the entries of $\mathbf{p}$ as a weight indicating the importance of a particular state to the performance of the MPC. We then sort these values in descending order keeping track of the indices, *i.e.* we generate the ordered list $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_{mH}]$ such that $p_{\alpha_1} \leq p_{\alpha_2} \leq \ldots \leq p_{\alpha_{mH}}$. Next we remove those values that correspond to the very last element of a control input sequence. This is to enforce the rule that blocking can only be done within a single control input sequence, *e.g.* we cannot block $u_1(H-1)$ and $u_2(0)$ as this would not make physical sense. We construct a sequence of increasingly blocked move blocking matrices beginning with the identity (no blocking) by moving along the list $\alpha$ and enforcing a single blocking move at a time via an elemental blocking matrix. We do this while being careful not to overlap any previous blocking moves using a counter variable $c$.

**Algorithm 1** Compute a Sequence of Move Blocking Matrices

**procedure** MBSeqGen($\mathbf{U}_r, m, H$)
  $\mathbf{p} \leftarrow \text{diag}(\mathbf{U}_r\mathbf{U}_r^T)$
  $\alpha \leftarrow \text{sort}(\mathbf{p})$
  $\alpha \leftarrow \alpha \setminus \{H, 2H, \ldots, mH\}$
  $\mathbf{M}^1 \leftarrow \mathbf{1}_{mH \times mH}$
  **for** $j$ from 1 to $m(H-1)$ **do**
    $c \leftarrow 0$
    **for** $k$ from 1 to $j-1$ **do**
      **if** $\alpha_k < \alpha_j$ **then**
        $c \leftarrow c + 1$
    $\mathbf{M}^{j+1} \leftarrow \mathbf{M}^j\mathbf{E}(mH - (j-1); \alpha_j - c)$
  **return** $[\mathbf{M}^1, \ldots, \mathbf{M}^{m(H-1)+1}]$

We refer to a MBrMPC that utilizes one of the $\mathbf{M}^j$ as an informed MBrMPC (IMBrMPC). By construction we know that $\mathbf{M}^{m(H-1)+1} = \bigoplus_{k=1}^{m} \mathbf{1}_H \in \mathbb{R}^{mH \times m}$ but what is relevant is the ordered sequence of increasingly move blocked matrices. Somewhere along this path the IMBrMPC will fail to have adequate performance. We want to find the

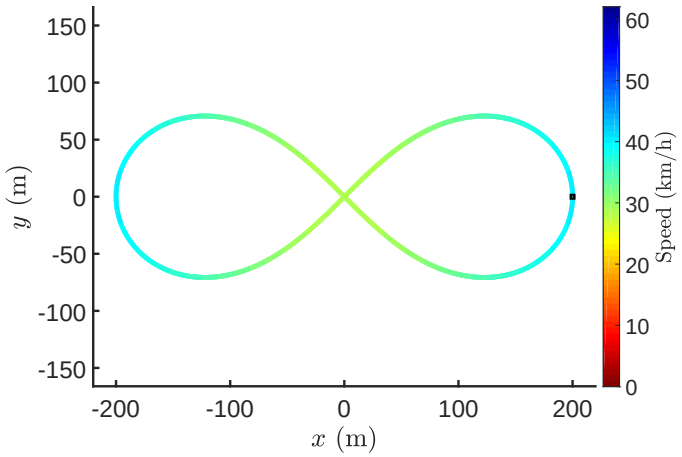



Figure 4.16: Reference trajectory for snapshot generation. Reference starts and ends at black square.

largest index $j$ such that the IMBrMPC corresponding to strategy $\mathbf{M}^j$ runs with acceptable performance. Currently this is done using trial and error but future work hopes to improve this through improved snapshot analysis.

### 4.3.3 Autonomous Vehicle Case Study

This problem is a reference tracking problem more general than the previous. We consider the fully nonlinear vehicle dynamics (see Appendix A for details) as well as realistic driving scenarios, such as lane changes, cornering and braking/accelerations.

For this problem we used the same cost function as the LMPC (6.2) except $\tilde{\mathbf{x}}(k) = f \circ^k (\mathbf{x}(0), \mathbf{U})$ where $f(\mathbf{x}, \mathbf{u}) = \mathbf{x} + \Delta t \Phi(\mathbf{x}, \mathbf{u})$, (a forward Euler discretization of the model), $H = 10$, $\mathbf{R} = \text{diag}(500, 0, 200)$ and the reference trajectory is of the form $\mathbf{x}^r = [x^r, y^r, v^r, 0, \psi^r, 0, 0]^T$. In this scenario we set $H = 10$ since it was the minimal horizon length needed for the original NMPC to satisfactorily track the selected reference trajectories. In all simulations the disturbances are $D_x, D_y \sim 10^2 \times \mathcal{N}$.

In Fig. 4.16 we display the reference trajectory used to generate our snapshot data. It is a gentle figure eight maneuver that takes $\approx$ 110s to complete. In Fig. 4.17 we display a selected sample of realistic urban driving maneuvers to test the PODrNMPC and IMBrNMPC.

In Figs. 4.18 and 4.19 we display results from simulation of the original NMPC, PO-

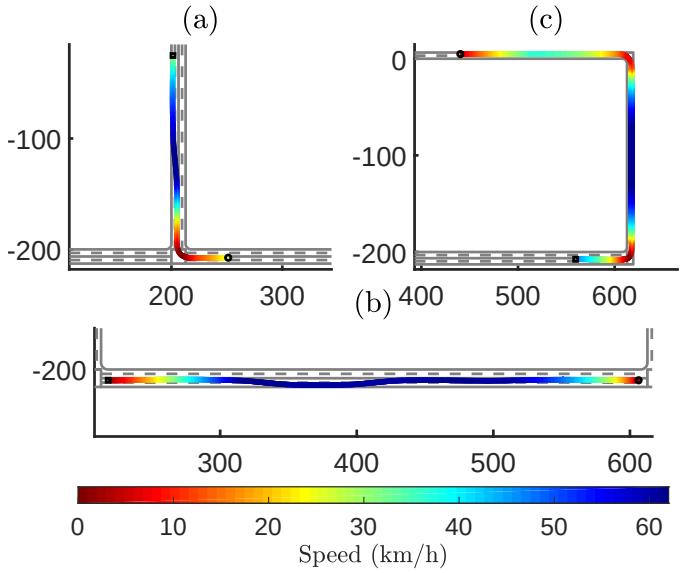



Figure 4.17: A selected sample of reference trajectories: (a) lane change followed by turn, (b) double lane change with initial and final acceleration, (c) turns with straights. All begin at the black square and end at the black circle

DrNMPC with $r = 4$ and IMBrNMPC with $r = 4$. These results demonstrate that even with a mismatch in snapshot, significant dimensional reduction (30 states to 4 states) can be found using the restricted Lagrangian approach presented here. The reference trajectory of the snapshot generation is not as aggressive as the tested maneuvers.

In Table 4.2 we summarize the relative tracking performance of the controllers. The unreduced NMPC results are denoted with $^*$. We take $\Delta w = 3.2$m as the lane width with which we measure relative position tracking error. The largest tracking error we see is in scenario (c) where the position tracking error is at worst 30.28cm for the standard NMPC, 31.21cm in the PODrNMPC case and 52.11cm for the IMBrNMPC.

To understand the role of $r$ on the TATs in Fig. 4.20 we display the TAT results of scenario (a). Firstly, we see that the acceleration factor of PODrNMPC is not as great as PODrLMPC. This is because of an increase in cost of computing the derivatives meaning the relative gain from reducing the linear solve step is less. Further the results of IMBrNMPC and PODrNMPC without a symbolic linear solver are very similar for equal values of $r$. The greatest gain in TAT speedup is made by combining IMBrNMPC with a symbolic linear solver. This is expected since the symbolic solver can take full advantage of eliminating redundant operations, *e.g.* carrying terms multiplied by 0 of which there are

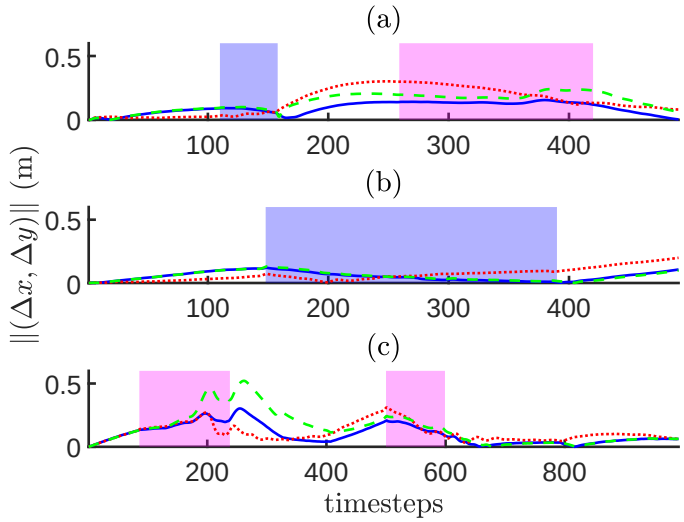



Figure 4.18: Original NMPC in blue (solid), PODrNMPC with $r = 4$ in red (dash) and IMBrNMPC with $r = 4$ in green (long dash). The blue patches highlight lane changes and red patches highlight turning maneuvers.

many since the move blocking matrix is full of zeros. For $r = 4$ this is an acceleration of 3.6× for IMBrNMPC and only 1.6× for PODrNMPC. Further we can see one advantage of the IMB approach versus standard move blocking. For a standard move blocking scheme (where all controls are similarly blocked) the smallest dimension for which we could get acceptable performance is $r = 6$, since we know $r = 3$ doesn't have good performance. However, at this value of $r$ the TAT acceleration decreases significantly.

In Table 4.3 we compare the tracking performance of the informed move blocking scheme to the optimal choice of move blocking strategy for $r = 4$ and our informed move blocking scheme to the optimal standard move blocking strategy for $r = 6 = 2m$ for the selected sample trajectories. For a fixed $r$ there are $\binom{m(H-1)}{r-m}$ possible move blocking strategies and for $r = qm, q \in \mathbb{N}, \binom{H-1}{q-1}$ standard move blocking strategies. In Table 4.3 we can see that for $r = 4$ the IMBrNMPC is close to optimal or actually optimal out of the 27 possible strategies. And further the IMBrNMPC strategy is very similar to the optimal MBrNMPC strategy as can be seen from the strategies shown in brackets. To demonstrate one advantage of the flexibility of our informed move blocking approach we compare the IMBrNMPC to the optimal standard move blocking strategy (SMBrNMPC) (where all controls are blocked similarly) for $r = 6$. We see with the exception of trajectory (b) the IMBrNMPC outperforms the optimal SMBrNMPC.

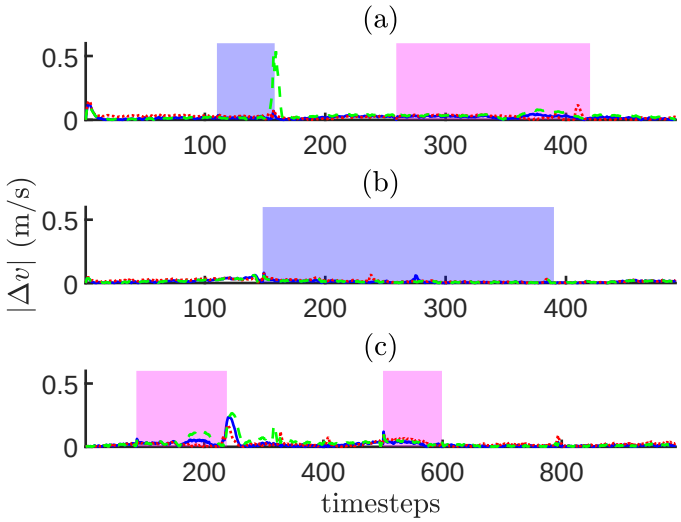



Figure 4.19: Original NMPC in blue (solid), PODrNMPC with $r = 4$ in red (dash) and IMBrNMPC with $r = 4$ in green (long dash).

## 4.4 Discussion

In this chapter we have presented a method to speedup the online optimization of MPCs. This method operates by extracting features of a sample set of MPC solutions to reduce the dimension of the problem Lagrangian. We propose two ways to do this: 1. a direct POD-Galerkin approach (PODrMPC) and 2. an informed move blocking strategy (IMBrMPC). From the results shown we can see that both strategies can provide an accelerated turnaround time with small tradeoff in controller performance.

We have utilized symSS to formulate our problems, which is a symbolics oriented approach.This approach also integrates well with PODrMPC when the reduced dimension $r$ is small (*i.e.* $< 7$) since we can turn the linear solve step into an optimized sequence of exact expressions, which accelerates an MPC further.

The initial step in creating a restricted Lagrangian is constructing the snapshot matrix. We propose collecting this data using a collection of representative controller simulations; however, what counts as representative for a given control problem may or may not be easy to identify. This problem of how to select the best snapshot matrix remains open. The preliminary results shown here demonstrate that the reduced MPCs are reasonably robust to the choice of snapshot, however, it would still be advantageous to develop methods to understand the limits of applicability of a snapshot. This problem is made difficult

Table 4.2: Relative Tracking Error

| | | $\max \Delta v/v^r - \max \Delta v^*/v^r$ | $[\max \|(\Delta x, \Delta y)\| - \max \|(\Delta x^*, \Delta y^*)\|]/\Delta w$ |
|---|---|---|---|
| PODrNMPC | (a) | 0.96% | 4.58% |
| | (b) | 0.13% | 2.44% |
| | (c) | −0.90% | 0.29% |
| IMBrNMPC | (a) | 1.93% | 2.67% |
| | (b) | 0.02% | 0.32% |
| | (c) | 0.04% | 6.82% |

Table 4.3: $\max \|(\Delta x, \Delta y)\|$ (cm)

| | NMPC | MBrNMPC | | | |
|---|---|---|---|---|---|
| | | $r = 4$ | | $r = 6$ | |
| | | IMBrNMPC[rank] ($[10, 10, 1, 9]$) | Optimal MBrNMPC | IMBrNMPC ($[1, 9, 10, 1, 1, 8]$) | Optimal SMBrNMPC |
| (a) | 15.51 | 24.06[$2^{\text{nd}}$] | 21.78($[10, 10, 2, 8]$) | 18.64 | 21.53($[2, 8]^3$) |
| (b) | 12.09 | 13.11[$5^{\text{th}}$] | 11.10($[10, 10, 3, 7]$) | 13.53 | 11.07($[3, 7]^3$) |
| (c) | 30.28 | 52.11[$1^{\text{st}}$] | – | 37.35 | 51.33($[1, 9]^3$) |

due to the interaction of problem nonlinearities and the sensitivity of the singular value decomposition.

We used trial and error to select the reduced dimension $r$ of PODrMPC and IMBrMPC. Our results indicate that for some small $r$ the PODrMPC performance does not decrease further meaning one can get away with relatively few tests in order to find a desired reduced dimension. It would be desirable to improve this situation by finding some function of the snapshot so no testing would be necessary. It appears that the level of reduction possible is sensitive to the presence of non-trivial entries in the rows of $\mathbf{U}_r$ and not the actual subspace itself. Further work is necessary to clarify this but it does indicate that the difficulty of bounding perturbed singular vectors may be sidestepped. A threshold on the bound $\max \tilde{B}_r^{\perp}$ appears to be a promising starting point.

The goal of this work has been to introduce a method which can reduce the TATs of MPCs while maintaining reasonable controller performance. We have focused here primarily on reducing the computational complexity of the linear solve step within an NLP solver by dimensional reduction. TAT accelerations $> 2\times$ were observed in our simulation

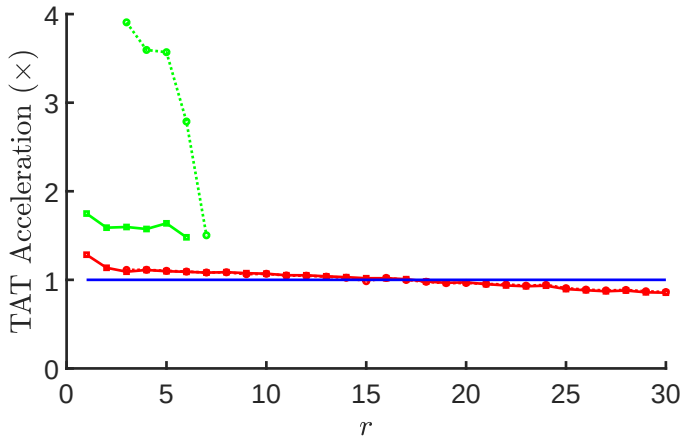



Figure 4.20: Role of $r$ in PODrNMPC and IMBrNMPC TAT acceleration. The results of PODrNMPC is given by solid lines with square points and the results of IMBrNMPC is given by dashed lines with circular points. The standard approach is given in red and the approach utilizing a symbolic linear solver are given in green. The standard NMPC is given by the solid blue line at 0.4826ms. These results are the average of the mean TAT of 100 simulations.

studies utilizing the strategies presented. However, we note that further TAT speedup is possible if we were able to reduce the computational cost of the derivatives themselves and not only a POD-Galerkin projection to simply reduce their dimension. The dimensional reduction we observed was significant and allowed the inclusion of a symbolic linear solver for further TAT acceleration. It should be noted that this dimensional reduction is not only useful for accelerating TATs but is useful for building models of controllers that are simple to compute. These models of controllers can be incorporated in other modules, *e.g.* integration of motion planning and control [60].

The strategies PODrMPC and IMBrMPC can be understood as an application of feature projection, from machine learning, to the problem of controller reduction. It is the authors' hope that more advanced machine learning methods can be applied in future, with fruitful results, to this problem. Future work also includes addressing theoretical issues surrounding this approach, *e.g.* does PODrMPC preserve stability or controllability of an MPC? Overall, this approach is applicable to a wide variety of problems but suffers from a lack of hard analysis.

# Chapter 5

# Nonlinear MPC Reduction Using Truncated Single-Shooting

In this chapter we present a scheme to reduce the computational burden of Nonlinear Model Predictive Controllers (NMPCs) when using the symSS approach. We identify the timestep as a 'small' parameter and conduct a perturbative analysis of Newton's step. This analysis leads to the introduction of a truncated Lagrangian that can be used to form a new quasi-Newton method with a symbolically reduced Hessian for online optimization.

This approach leverages the power of symbolic computation to generate efficient controller code for online evaluation. The reduction methods were applied to diesel engine control and electric vehicle cruise control problems and achieved turnaround times more than 2 times faster with no tradeoff in controller performance. These initial results are quite promising for the development of real-time NMPCs and introduce some new avenues for research.

The receding horizon principle behind NMPC is general enough that there exist multiple formulations [61]. In this chapter we focus on the single shooting (SS) (sometimes called control parameterization) approach where only the control inputs, and not the states, are exposed as degrees of freedom in the transcribed optimization problem [62]. This approach yields a denser optimization problem as compared to collocation, the more popular approach, which treats plant dynamics as equality constraints. SS has been used in trajectory planning and vehicle collision avoidance problems [63, 64] but has typically been avoided since it suffers from poor stability properties [14]. In its numerical implementation, SS has two stages. The model is first simulated in an inner loop and then the derivatives are computed, through sensitivity or adjoint equations, for the optimizer that runs in an

outer loop. Such computations are expensive and this staged approach, though straightforward to implement, is often outperformed by other methods [65]. In this chapter we leverage the power of symbolic computation for controller reduction and code generation so that our controllers operate efficiently in a single stage using symSS.

To achieve efficient controllers, recent work has focused on direct collocation with specially designed solvers to take advantage of the shape and sparsity of the underlying equations [18, 17]. In spirit, the work of this paper is opposite to this trend in that we purposefully generate a dense system of minimal dimension. In this way most of the cost is not in the solution but the computation of the system to be solved, which we simplify using optimized code generation and our truncation strategy.

In this chapter we consider a simple tracking problem of the form

$$\begin{aligned} \min_{\mathbf{U}} \left\{ \mathcal{J}(\mathbf{U}; \mathbf{x}(0)) = \frac{1}{2} \left( \sum_{k=0}^{H-1} \|\Delta \mathbf{x}(k+1)\|_{\mathbf{Q}}^2 + \sum_{k=0}^{H_c-1} \|\mathbf{u}(k)\|_{\mathbf{R}}^2 \right) \right\} \\ \text{subject to} \quad 0 \leq \mathbf{h}(\mathbf{x}(k), \mathbf{u}(k)) \quad k = 0, \ldots, H-1 \end{aligned} \tag{5.1}$$

where $\|\cdot\|_{\mathbf{A}}^2$ denotes the weighted vector norm of matrix $\mathbf{A}$, $\mathbf{Q} \in \mathbb{R}^{n \times n}, \mathbf{R} \in \mathbb{R}^{m \times m}$ are positive definite matrices selected to balance the tracking error and control cost, $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}^r$ and $\mathbf{x}^r$ is a reference trajectory. For computational simplicity the states over the prediction horizon are defined using a one-step method $\Psi$ to discretize the plant model $\dot{\mathbf{x}} = \phi(\mathbf{x}, \mathbf{u})$ via

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \Delta t \Psi(\mathbf{x}(k), \mathbf{u}(k)). \tag{5.2}$$

The simplest one-step method is the forward Euler scheme where $\Psi = \phi$.

The Lagrangian of (5.1) is

$$\mathcal{L}(\mathbf{U}) = \Theta(\mathbf{U}) + \mathcal{J}(\mathbf{U}) \tag{5.3}$$

where $\Theta$ contains any terms used to enforce the constraints *e.g.* any Lagrange multiplier terms and barrier or penalty functions. Typically $\Theta$ can be expressed as a sum of the constraints over the prediction horizon and is of the form

$$\Theta(\mathbf{U}) = \sum_{i=1}^{q} \sum_{k=0}^{H-1} \theta_i(h_i(\mathbf{x}(k), \mathbf{u}(k))) \tag{5.4}$$

where function $\theta_i$ captures the method of handling constraint $h_i$ and $q$ is the total number of constraints.

## 5.1 Perturbative Expansion of Single Shooting

If the underlying plant model $f$ is Lipschitz continuous in $\mathbf{x}$ with Lipschitz constant $L$ then $\Delta t = C/L$ for some constant $C$ [66, 67]. Furthermore, if $C$ is small or, if over the control scenario of interest the rate of change of the model does not approach $L$, then we can identify $\Delta t$ as a 'small' parameter. Thus we can expect to reduce the computational cost of the linear solve (2.5) by conducting a perturbative analysis to truncate the expressions without a significant increase in solution error.

For ease of presentation we take a closer look at a simple one-dimensional system $\dot{x} = f(x, u)$ with $x, u \in \mathbb{R}$ and $H_c = H_p = H$. For notational compactness in this section, we denote $x(k) = x^k, u(k) = u^k$ so that $\mathbf{U} = \left(u^0 \cdots u^{H-1}\right)^T$. We begin by examining the cost term $\mathcal{J}$ which can be written as

$$\mathcal{J}(\mathbf{U}; x(0)) = \frac{1}{2} \sum_{k=0}^{H-1} Q(\Delta x^{k+1})^2 + R(u^k)^2$$

where $Q, R \in \mathbb{R}$. To obtain $\nabla \mathcal{J}$ and $\mathbf{H}_{\mathcal{J}}$ we need to compute terms $\frac{\partial \mathcal{J}}{\partial u^j}$ and $\frac{\partial^2 \mathcal{J}}{\partial u^{j+m} \partial u^j}$ for $m \geq 0$, which require the derivatives of the states with respect to the controls, *e.g.* $\frac{\partial x^{k+1}}{\partial u^j}$. The states in symSS are defined recursively by (5.2) and so the derivatives are also defined recursively, *e.g.*

$$\frac{\partial x^{k+1}}{\partial u^j} = \left(1 + \Delta t \frac{\partial \Psi}{\partial x}\bigg|_k\right) \frac{\partial x^k}{\partial u^j} + \Delta t \frac{\partial \Psi}{\partial u}\bigg|_k \delta_{j,k},$$

and

$$\begin{aligned}\frac{\partial^2 x^{k+1}}{\partial u^{j+m} \partial u^j} = \left(1 + \Delta t \frac{\partial \Psi}{\partial x}\bigg|_k\right) \frac{\partial^2 x^k}{\partial u^{j+m} \partial u^j} + \Delta t \left(\frac{\partial^2 \Psi}{\partial x^2}\bigg|_k \frac{\partial x^k}{\partial u^{j+m}} + \frac{\partial^2 \Psi}{\partial u \partial x}\bigg|_k \delta_{j+m,k}\right) \frac{\partial x^k}{\partial u^j} \\ + \Delta t \left(\frac{\partial^2 \Psi}{\partial x \partial u}\bigg|_k \frac{\partial x^k}{\partial u^{j+m}} + \frac{\partial^2 \Psi}{\partial u^2}\bigg|_k \delta_{j+m,k}\right) \delta_{j,k}\end{aligned}$$

where $\frac{\partial \Psi}{\partial x}\big|_k$ denotes $\frac{\partial \Psi}{\partial x}(x^k, u^k)$ and $\delta_{j,k}$ is the Kronecker delta. If we expand the first and second derivatives of the states

$$\frac{\partial x^{k+1}}{\partial u^j} = \begin{cases} 0 & j > k \\ \Delta t \frac{\partial \Psi}{\partial u}\Big|_j & j = k \\ \Delta t \frac{\partial \Psi}{\partial u}\Big|_j \prod_{l=1}^{k-j} \left(1 + \Delta t \frac{\partial \Psi}{\partial x}\Big|_{k+1-l}\right) & j < k \end{cases}$$

and,

$$\frac{\partial^2 x^{k+1}}{\partial u^{j+m}\partial u^j} = \begin{cases} 0 & j > k \\ \Delta t \frac{\partial^2 \Psi}{\partial u^2}\Big|_j \delta_{m,0} & j = k \\ \Delta t \frac{\partial^2 \Psi}{\partial u^2}\Big|_j \delta_{m,0} \prod_{l=1}^{k-j} \left(1 + \Delta t \frac{\partial \Psi}{\partial x}\Big|_{k+1-l}\right) & \\ +\Delta t^2 \frac{\partial \Psi}{\partial u}\Big|_j \sum_{p=1}^{k-j} \prod_{\substack{l=1 \\ l\neq p}}^{k-j} \left(1 + \Delta t \frac{\partial \Psi}{\partial x}\Big|_{k+1-l}\right) \left(\frac{\partial^2 \Psi}{\partial x^2}\Big|_p \frac{\partial x^p}{\partial u^{j+m}} + \frac{\partial^2 \Psi}{\partial u \partial x}\Big|_p \delta_{j+m,p}\right) & j < k \end{cases}$$

and then truncate to first order in $\Delta t$ we uncover the simple expressions

$$\frac{\partial x^{k+1}}{\partial u^j} \approx \begin{cases} 0 & j > k \\ \Delta t \frac{\partial \Psi}{\partial u}\Big|_j & j \leq k \end{cases} \tag{5.5}$$

and

$$\frac{\partial^2 x^{k+1}}{\partial u^{j+m}\partial u^j} \approx \begin{cases} 0 & j > k \\ \Delta t \frac{\partial^2 \Psi}{\partial u^2}\Big|_j \delta_{m,0} & j \leq k \end{cases}. \tag{5.6}$$

This leads to the following expressions for the derivatives of the cost term to first order in $\Delta t$

$$\frac{\partial \mathcal{J}}{\partial u^j} \approx R u^j + \Delta t \frac{\partial \Psi}{\partial u}\Big|_j \sum_{k=j}^{H-1} Q \Delta x^{k+1}, \tag{5.7}$$

$$\frac{\partial^2 \mathcal{J}}{\partial u^{j+m}\partial u^j} \approx \delta_{m,0} \left(R + \Delta t \frac{\partial^2 \Psi}{\partial u^2}\Big|_j \sum_{k=j}^{H-1} Q \Delta x^{k+1}\right). \tag{5.8}$$

Notably (5.8) indicates that $\mathbf{H}_{\mathcal{J}}$ is diagonally dominant and that all off-diagonal terms are of order $\Delta t^2$ or higher.

Turning our attention to the constraint term, if we assume $\Theta$ is of the form given by (5.4), then $\Theta$ only weakly couples the controls from different timesteps. Using similar

analysis to the above we find, to first order in $\Delta t$,

$$\frac{\partial \Theta}{\partial u^j} \approx \sum_{i=1}^{c} \left( \frac{\partial \theta_i}{\partial g} \frac{\partial g_i}{\partial u}\bigg|_j + \Delta t \frac{\partial \Psi}{\partial u}\bigg|_j \sum_{k=j+1}^{H-1} \frac{\partial \theta_i}{\partial g} \frac{\partial g_i}{\partial x}\bigg|_k \right),$$

$$\frac{\partial^2 \Theta}{\partial u^{j+m} \partial u^j} \approx \sum_{i=1}^{c} \left( \delta_{m,0} \left[ \frac{\partial^2 \theta_i}{\partial g^2} \left( \frac{\partial g_i}{\partial u}\bigg|_j \right)^2 + \frac{\partial \theta_i}{\partial g} \frac{\partial^2 g_i}{\partial u^2}\bigg|_j + \Delta t \frac{\partial^2 \Psi}{\partial u^2}\bigg|_j \sum_{k=j+1}^{H-1} \frac{\partial \theta_i}{\partial g} \frac{\partial g_i}{\partial x}\bigg|_k \right] \right.$$
$$\left. + (1-\delta_{m,0}) \Delta t \frac{\partial \Psi}{\partial u}\bigg|_j \left[ \frac{\partial^2 \theta_i}{\partial g^2} \frac{\partial g_i}{\partial u}\bigg|_{j+m} \frac{\partial g_i}{\partial x}\bigg|_{j+m} \frac{\partial \theta_i}{\partial g} \frac{\partial^2 g_i}{\partial u \partial x}\bigg|_{j+m} \right] \right).$$

From this we can see that $\mathbf{H}_\Theta$ is also diagonally dominant because off-diagonals of order $\Delta t$ are present only if some of the constraints in $\mathbf{g}$ involve both states and controls.

These observations can be used to define a truncated Lagrangian whose gradient and Hessian are equal to the original Lagrangian to first order in $\Delta t$. Uncovering such an expression allows us to easily generate code for the truncated system using symbolic computing software like MAPLE. For the general problem (5.1), the truncated Lagrangian is

$$\mathcal{L}^t(\mathbf{U}, \mathbf{V}; \mathbf{x}(0)) = \Theta^t(\mathbf{U}, \mathbf{V}; \mathbf{x}(0)) + \mathcal{J}^t(\mathbf{U}, \mathbf{V}; \mathbf{x}(0)) \tag{5.9}$$

where

$$\Theta^t(\mathbf{U}, \mathbf{V}; \mathbf{x}(0)) = \sum_{i=1}^{c} \sum_{k=0}^{H_p-1} \theta_i(g_i(\mathbf{x}(k), \mathbf{u}(k)) + \Delta t \Psi(\mathbf{x}(k), \mathbf{u}(k))^T \Gamma(\mathbf{V}, k; \mathbf{x}(0)) \tag{5.10}$$

with $\Gamma(\mathbf{V}, k; \mathbf{x}(0)) = \sum_{l=k+1}^{H_p-1} \frac{\partial \theta_i}{\partial g} \frac{\partial g_i}{\partial \mathbf{x}}\Big|_l$ and

$$\mathcal{J}^t(\mathbf{U}, \mathbf{V}; \mathbf{x}(0)) = \sum_{k=0}^{H_p-1} \Delta t \Psi(\mathbf{x}(k), \mathbf{u}(k))^T \mathbf{Q} X(\mathbf{V}, k; \mathbf{x}(0)) + \frac{1}{2} \sum_{k=0}^{H_c-1} \|\mathbf{u}(k)\|_{\mathbf{R}}^2, \tag{5.11}$$

with $X(\mathbf{V}, k; \mathbf{x}(0)) = \sum_{l=k}^{H_p-1} \Delta \mathbf{x}(l+1)$. It is critical to note that in (5.10) and (5.11) the states $\mathbf{x}(k)$ are realized as functions of $\mathbf{V}$ not $\mathbf{U}$. Further, in expressions $\Gamma, X$ any arguments $\mathbf{u}$ have been replaced by the dummy variable $\mathbf{v}$, *e.g.* $\frac{\partial g_i}{\partial \mathbf{x}}\Big|_1 = \frac{\partial g_i}{\partial \mathbf{x}}(\mathbf{x}(1), \mathbf{v}(1))$ where $\mathbf{x}(1) = \mathbf{x}(0) + \Delta t \Psi(\mathbf{x}(0), \mathbf{v}(0))$. With this definition of the truncated Lagrangian, $\nabla \mathcal{L}^t$ and $\mathbf{H}_{\mathcal{L}^t}$

taken with respect to the first argument $\mathbf{U}$ and evaluated at $\mathbf{V} = \mathbf{U}$ are equal to $\nabla\mathcal{L}$ and the diagonals of $\mathbf{H}_{\mathcal{L}}$ to first order in $\Delta t$, respectively. The dummy variable $\mathbf{V}$ has been introduced to allow for easy symbolic differentiation since we can apply automatic differentiation of $\mathcal{L}^t$ with respect to $\mathbf{U}$ followed by the substitution $\mathbf{V} \leftarrow \mathbf{U}$.

The fact that the truncated system is block diagonal is not only good for an efficient linear solution but can be exploited further. How this can be done is discussed in the next section where we introduce a compression scheme.

## 5.2 Truncated and Compressed Single Shooting

Truncated Single Shooting (TSS) for NMPC can be implemented by replacing $\mathbf{H}_{\mathcal{L}}(\mathbf{U})$ in (2.5) with $\mathbf{H}_{\mathcal{L}^t}(\mathbf{U}, \mathbf{V})$ where the derivatives have been taken with respect to the first argument and are evaluated at $\mathbf{V} = \mathbf{U}$. It was found that also replacing $\nabla\mathcal{L}$ led to worse controller performance with no significant computational savings. In this regard TSS can be seen as a quasi-Newton method for SS with a block diagonal reduced Hessian.

The perturbative analysis of Section 5.1 suggests that $\mathbf{H}_{\mathcal{L}}$ is block diagonally dominant meaning the solutions $\mathbf{u}^*(k)$ for different timesteps $k$ in the horizon are weakly coupled. Since the NMPC algorithm implements $\mathbf{u}^*(0)$ we can take advantage of this and only solve for $\mathbf{u}^*(0)$ with minimal degradation in controller performance. This can be done with or without truncation.

A compressed system is formed by simply changing the input argument of the problem Lagrangian to form a compressed Lagrangian

$$\mathcal{L}^c(\mathbf{u}'; \mathbf{x}(0)) = \mathcal{L}(\mathbf{U}^c; \mathbf{x}(0)) \tag{5.12}$$

where $\mathbf{U}^c = \begin{pmatrix}\mathbf{u}'^T & \mathbf{u}^c(1)^T & \cdots & \mathbf{u}^c(H_c-1)^T\end{pmatrix}^T$. The resulting linear system

$$\mathbf{H}_{\mathcal{L}^c}(\mathbf{u}')\Delta\mathbf{u}'_{NS} = -\nabla\mathcal{L}^c(\mathbf{u}')^T \tag{5.13}$$

where the gradient and Hessian have been derived with respect to $\mathbf{u}'$. Equation (5.13) is of minimal dimension and its solution $\mathbf{u}'^*$ is often a remarkably good approximation to the actual $\mathbf{u}^*(0)$ when using Newton's method. The compressed system minimizes the expense of the linear solve by reducing the problem dimension from $mH_c$ to $m$. A SS NMPC that uses (5.13) instead of (2.5) is called a compressed single shooting (CSS) NMPC controller, and when compression is combined with truncation we call it truncated compressed single shooting (TCSS).

There is one important design decision to be made when utilizing CSS. Although we need to solve for only the next control step, we need to use some sequence of control inputs over the horizon to compute the gradient and Hessian, that is, we need to choose the $\mathbf{u}^c(k)$ values. Depending on the control objective and the dynamics of the plant, this can be done a number of different ways. One would be to set all $\mathbf{u}^c(k) = \mathbf{u}'$ which is equivalent to simply setting $H_c = 1$. Other options are to use a fixed equilibrium control input or to use $\mathbf{u}'$ from the previous timestep. Many other strategies could be used but they would all require using a simple rule with values from previous timesteps, precomputed values or guesses since the $\mathbf{u}^c(k)$ values are never updated directly from (5.13). One potential direction for future work would be utilizing the feature projection idea used in Chapter 4 for this task. One detail to note is that, although we include the case $H_c = 1$ in CSS, it differs from most other strategies since the computational complexity of the derivatives increases as $\mathbf{u}^c(k) = \mathbf{u}'$ for all timesteps $k$ in the prediction horizon.

## 5.3 Case Studies

In this section we apply truncation and compression to two control scenarios. The first is a diesel airpath (DAP) controller and the second is an electric vehicle cruise control problem.

### 5.3.1 Implementation Details

Our (T/C)SS NMPCs were implemented using Newton's method for online optimization. To apply Newton's method, the functions for the gradient and Hessian were precomputed using MAPLE using the symSS formulation presented in Chapter 3. In this way each iteration of the solver only involved two computational steps: first, evaluating the gradient and Hessian at the current iteration and second, solving the linear system for the iteration update. Unlike numerical SS methods no simulation stage was necessary. We fixed the number of Newton iterations at 1 and used a precomputed warm start control for the initial step.

### 5.3.2 Diesel Airpath Control

The main goal of this controller is to track a reference output trajectory where $\mathbf{y} = \begin{pmatrix} p_{int} & \Psi_{EGR} \end{pmatrix}^T$, the intake manifold pressure and exhaust gas recirculation (EGR) rate the outputs of interest. Tracking these two outputs allows the engine to meet the driver's

demands while satisfying emission requirements. The DAP model under consideration is a 9 state, nonlinear, physics based-model with 3 controls and 2 disturbance inputs. More information on the model and its derivation can be found in [53] and the references therein.

A further objective of the controller is to minimize the rate of change of the controls $\mathbf{u} = \begin{pmatrix} \theta & u_{EGR} & u_{VGT} \end{pmatrix}^T$, throttle command [% closed], EGR valve command [% open], and variable-geometry turbocharger (VGT) command [% normalized], while satisfying the control bounds $\theta \in [0, 100], u_{EGR} \in [0, 70]$ and $u_{VGT} \in [40, 80]$. The cost is given by

$$\mathcal{J}(\mathbf{U}; \mathbf{x}(0)) = \sum_{k=0}^{H_p-1} \|\Delta \mathbf{y}(k+1)\|_{\mathbf{Q}}^2 + \sum_{k=0}^{H_c-1} \|\Delta \mathbf{u}(k)\|_{\mathbf{R}}^2$$

where $\Delta \mathbf{u}(k) = \mathbf{u}(k+1) - \mathbf{u}(k)$, $\mathbf{Q} = \text{diag}\begin{pmatrix} 1 & 10^4 \end{pmatrix}$ and $\mathbf{R} = \text{diag}\begin{pmatrix} 10 & 10 & 10 \end{pmatrix}$. The control bounds were enforced using the penalty function

$$\Theta(\mathbf{U}) = \sum_{k=0}^{H_c-1} \|\rho_p(\mathbf{u}(k))\|_{\mathbf{P}}^2 \tag{5.14}$$

where the half-penalty function $\rho_p$ is the penalty function given in Section 3.1. In our simulations we set $p = 4$ and $\mathbf{P} = \mathbf{I}$. We used a timestep of $\Delta t = 10$ms and prediction horizon $H_p = 10$. The disturbances were input as piecewise constants perturbed every 1.43s.

### Results

For the purposes of comparing controller turnaround times (TATs) and performance we considered 8 different controllers. Those simulations labeled $\text{SS}_k$ use the full Lagrangian of the single shooting method (5.3) and those labeled $\text{TSS}_k$ use Hessian of the truncated Lagrangian with $H_c = k$. Labels CSS and TCSS refer to the simulations with compressed and both truncated and compressed Lagrangians, respectively. In our simulations we choose the compression strategy of setting all $\mathbf{u}^c$ values equal to the control input of the previous timestep. All TATs were measured using a MicroAutobox II 1401 real-time computer. The TAT reduction factors are computed with respect to the SS controller with the same control horizon.

In Table 5.1 we summarize the results of the DAP control simulations where a measure of the controller performance over the simulation is given by $\tilde{J}_- = \sum_{k=0}^{999} \|\Delta \mathbf{y}(k+1)\|_{\mathbf{Q}}^2 + \|\widehat{\mathbf{u}}(k)\|_{\mathbf{R}}^2$. We can see that larger control horizons result in longer TATs but have better

Table 5.1: DAP Controller Performance

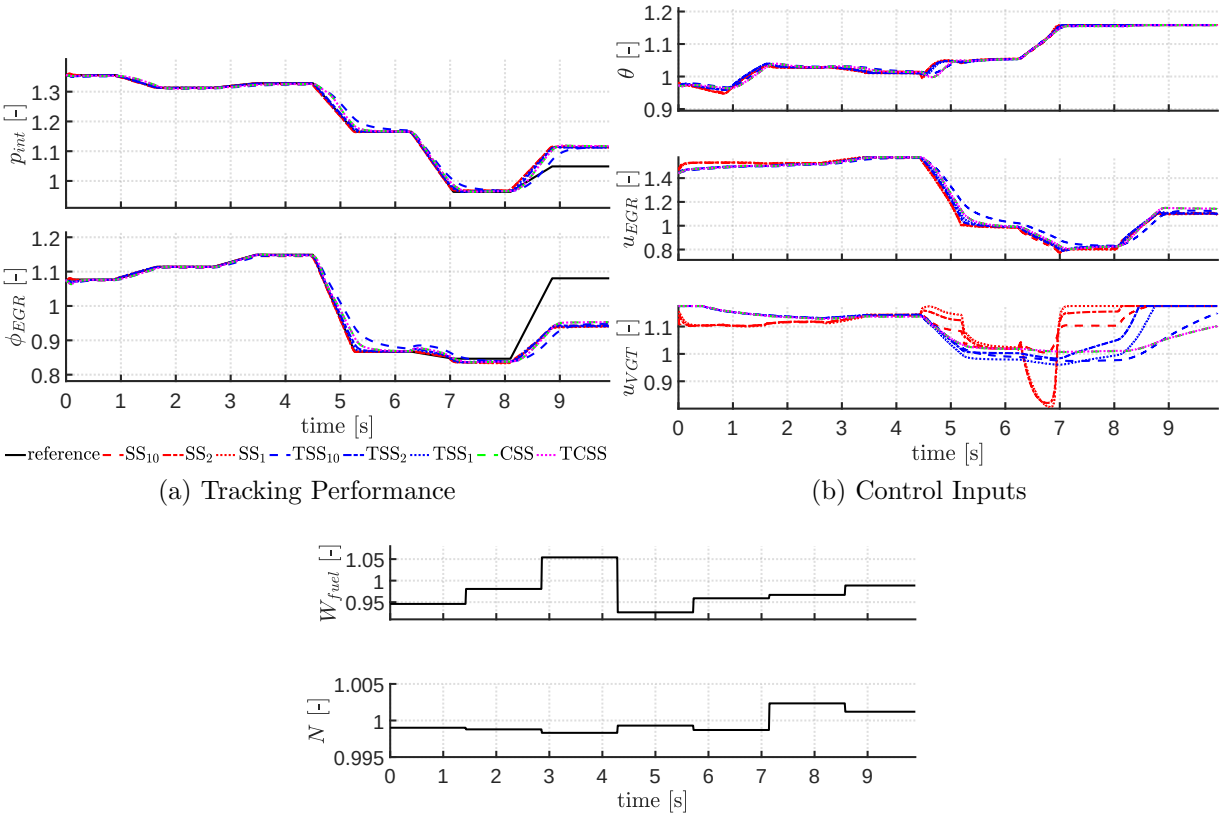
| | Max TAT [ms] | TAT Reduction [×] | $\tilde{J}_-/\tilde{J}_{\text{TCSS}}$ [–] |
|---|---|---|---|
| $SS_{10}$ | 0.523 | – | 0.992 |
| $TSS_{10}$ | 0.157 | 3.34 | 0.987 |
| $SS_2$ | 0.191 | – | 1.002 |
| $TSS_2$ | 0.094 | 2.04 | 0.998 |
| $SS_1$ | 0.139 | – | 1.005 |
| $TSS_1$ | 0.091 | 1.52 | 0.998 |
| CSS | 0.134 | 1.01 | 1.000 |
| TCSS | 0.067 | 2.06 | 1 |

performance, as expected. Further, the larger the control horizon the larger the potential there is for a reduction in TAT. As alluded to in Section 5.2 we see that $SS_1$ and CSS are about equal in TAT which is in line with the observation that one can view $SS_1$ as a compression strategy. The combination of truncation and compression does give us the best TAT. Interestingly we see that in terms of performance the truncated controllers actually performed slightly better than the non-truncated. This result is not expected but indicates that the reduction in TAT from these strategies came without any expense in terms of controller performance.

In Fig. 5.1 we display the simulation results of the different controllers. We can see that all controllers perform reasonably well over the first 8s (when tracking is lost due to a combination of engine regime and controller saturation) and that the CSS and TCSS solutions are effectively equal. Even though the truncated controllers have greater tracking error, they compensate by having more gentle control inputs. This is most notable in the results of $u_{VGT}$.

### 5.3.3 Electric Vehicle Cruise Control

The goal of this controller is to track a desired reference velocity given here by the SFTP US06 driving cycle. The vehicle model is a 6 state, stiff, nonlinear model representative of the Toyota RAV4 EV that incorporates tire slip dynamics [68]. The model captures only longitudinal vehicle dynamics and has states $\mathbf{x} = \begin{pmatrix} \omega_m & \omega_w & v & \theta_m & \theta_w & \lambda_t \end{pmatrix}^T$, which are motor angular speed, wheel angular speed, vehicle velocity, motor torsion angle, wheel angle and wheel slip, respectively. Further, there is a single control input, $\mathbf{u}$ = T [N




(a) Tracking Performance

(b) Control Inputs

(c) Disturbance Inputs

Figure 5.1: DAP controller simulation. Note: the displayed values have been normalized and given an offset.

m], motor torque. See [69] for more details about the model and a multi-objective cruise controller with slip-based constraints.

In our simulations we set $H_p = 10$ and $\Delta t = 0.1$ms. The cost function is the same as that in (5.1) with $\mathbf{Q} = \text{diag}\begin{pmatrix}0 & 0 & 2\times 10^9 & 0 & 0 & 0\end{pmatrix}$ and $\mathbf{R} = 2\times 10^{-3}$. The control constraint $\mathrm{T} \in [-350, 350]$ was enforced using the penalty function (5.14) with $\mathbf{P} = 10^3$.

### Results

In Table 5.2 we display the results of the cruise control simulations. The overall reduction factor in TAT was about 2× which is inline with our DAP control simulations. The

Table 5.2: Cruise Controller Performance

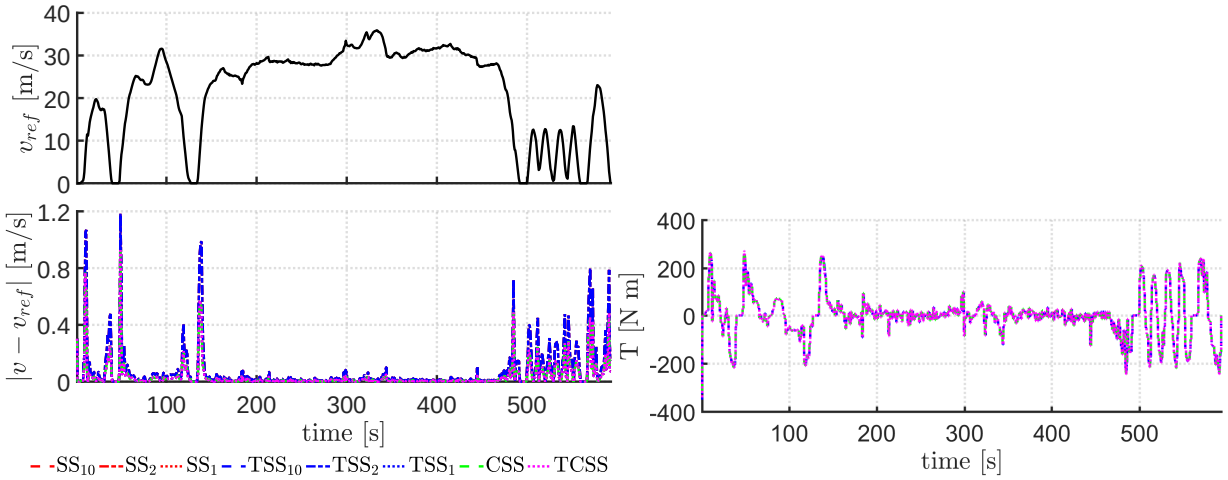
| | Max TAT [ms] | TAT Reduction [×] | $\max \lvert v - v_{ref} \rvert$ [m/s] |
|---|---|---|---|
| $SS_{10}$ | 0.0240 | – | 1.184 |
| $TSS_{10}$ | 0.0114 | 2.10 | 1.184 |
| $SS_2$ | 0.0130 | – | 1.184 |
| $TSS_2$ | 0.0062 | 2.08 | 1.184 |
| $SS_1$ | 0.0120 | – | 0.925 |
| $TSS_1$ | 0.0061 | 1.97 | 0.925 |
| CSS | 0.0121 | 1.00 | 0.925 |
| TCSS | 0.0062 | 1.95 | 0.925 |

slight differences in TAT reduction observed between our two control scenarios is due to the different underlying models. The DAP model is more computationally complex and thus has longer TATs but also greater TAT reduction using truncation. Unexpectedly, those simulations with shorter control horizons performed better in velocity tracking performance. Again we can conclude that the TAT reduction came at no expense to the controller performance.

In Fig. 5.2 we display the results of the cruise controller simulations. The tracking performance of all controllers is poorest when $v_{ref}$ approaches 0 but otherwise performs very well, *e.g.* over the period from 200s to 300s all the controllers are within 0.05% of $v_{ref}$. As we can see from the zoomed in views the controllers with a single control horizon perform better since the torque response is slightly faster.

## 5.4 Discussion

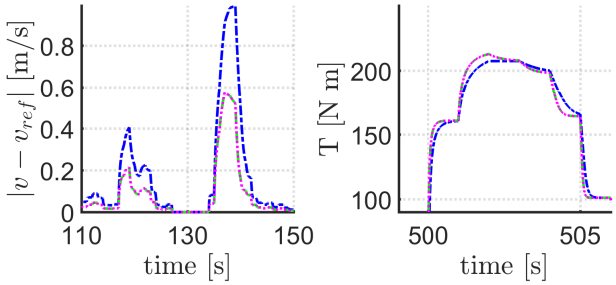
We were able to reduce TATs by half without a reduction in controller performance using the methods presented here. Truncation allowed us to reduce the computational complexity of the Hessian used in Newton's method for optimization and compression yielded further solver reductions. This approach is unique among quasi-Newton methods in that it is not a Hessian approximation but a reduction at the symbolic level of the exact Hessian. In our implementation we used the single shooting approach to NMPC and utilized symbolic computing software to precompute and optimize the required function calls necessary for Newton steps. This allowed us to eliminate a stage of computation typical to purely numerical implementations and to easily realize the truncation and compression schemes.




(a) Tracking Performance




(b) Control Inputs




(c) Zoomed in Views

Figure 5.2: Cruise controller simulations. Note that $SS_1$, $TSS_1$, CSS and TCSS are all displayed on top of one other and so too are $SS_{10}$, $TSS_{10}$, $SS_2$ and $TSS_2$.

This method is most appropriate for systems with relatively slow dynamics, in comparison to the timestep. Also, for those systems with models with computationally complicated expressions this truncation approach will lead to greatest TAT reduction.

# Chapter 6

# Towards Integrated Planning and Control of Autonomous Vehicles Using Nested MPCs

This chapter presents a strategy to integrate the planning and control for autonomous vehicles. This strategy takes advantage of the compact symSS formulation and the insights of Chapter 5. The aim of this work is to provide a method that can yield controller feasible reference paths, *i.e.* paths that are not only dynamically feasible but are feasible under the action of a low-level feedback controller. The method is designed to find a control feasible parameterization of a collision-free path provided by a path generation scheme, *e.g.* rapidly-exploring random trees or one of its many variants. This parameterization is found such that the vehicle under the action of the low-level controller will be able to follow that path within a specified tolerance. The design is based on a feedback strategy with nested MPCs for planning and control. The results presented here are preliminary but hint at the benefits of such a strategy and suggest avenues for future work.

## 6.1 Integrated Planning and Control: Progress

The integration of path planning and control for autonomous vehicles remains a challenging, safety-critical problem. Part of the challenge of this problem comes from the union of a multitude of constraints, *e.g.* vehicle dynamics, control bounds, and a collision-free trajectory, that must be synthesized to yield a robust solution method. What we require is a

method to generate robust collision-free reference trajectories that are guaranteed feasible under the action of the low-level feedback controller. The real-time demand of an actual vehicle and the unpredictability of the environment increases the challenge further.

In recent work this problem has been recognized as an important issue. Gao et al. state that "real-time generation and tracking of feasible trajectories is a major barrier in autonomous guidance systems" [70] and in a recent paper by Li et al. they state "a better communication or integration between the high-level motion planning and the low-level tracking control would be very valuable for enhancing the overall performance of [autonomous ground vehicles]" [71]. In a recent review Berntorp states "using feedback-based planning bridges the gap between the path-planning and vehicle-control modules, and can potentially offer a more integrated approach to path planning and collision avoidance in autonomous driving" [72]. The work presented here follows this advice by using a nested Model Predictive Control (MPC) design to integrate planning and control.

For the most part there have been two approaches to the problem of planning and control integration. The first is to integrate any environmental obstacles as constraints within the vehicle controller using MPC [73, 74, 75, 76, 77, 70]. This eliminates the need for a local path planner altogether since the obstacle avoidance and control is handled simultaneously. However, this approach significantly increases the complexity of the Nonlinear Program (NLP) within the controller for arbitrary environments which in turn increases the computational burden of the method. The integration is complete but makes it much harder to meet the real-time demand.

The second approach has been to split the problem into two parts or loops. The inner loop is a high-frequency low-level trajectory tracking controller and the outer loop is a lower frequency local path planner designed to handle the unpredictability of the environment. The process of understanding the environment (*e.g.* obstacle recognition and classification, collision-free path generation) is much more computationally demanding than solving a reference tracking problem over a short horizon. Thus this architecture takes advantage of the natural timescales of the problem. This two loop architecture was successfully used by Team MIT's entry for the 2007 DARPA Urban Challenge [78]. On their vehicle the motion planner ran at approximately 10Hz and the controller at 25Hz.

The challenge of this second approach is to integrate the processes. It is natural for the path planner to feed a reference trajectory to the low-level controller but *it remains a challenge to incorporate the controller into the path planner*. Most work to date incorporates vehicle kinematics or dynamics into the path generation stage but fails to integrate the controller behaviour [79, 80, 81, 82, 83, 84, 85, 86, 71]. *e.g.* in [79] the path is planned using a simulation based sampling method to include vehicle dynamics, in [80] they use

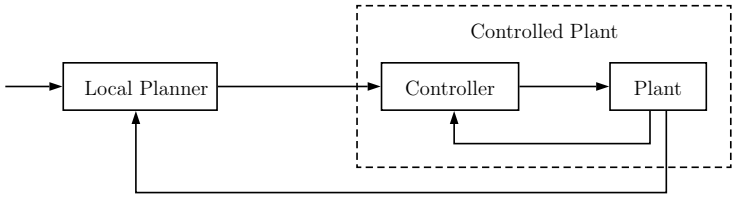



Figure 6.1: Planning and control feedback loops.

RRT integrated with a dynamic vehicle model and in [81] they use a kinematic model to design a path planner for collision avoidance and a dynamic model for the MPC controller. The framework proposed by [71] uses a point-mass kinematic model and control constraints to generate paths by solving a boundary value problem. In [78] they use the closed-loop rapidly-exploring random tree (CL-RRT) algorithm to make use of a low-level controller in the planning stage. This is the first instance, to the authors' knowledge, that the low-level controller behaviour has been integrated in the planning stage. The controller used in the planning stage is a pure-pursuit controller of kinematic bicycle model for steering and a PI controller for speed tracking. This is a rudimentary model of a controller that has the advantage of being analytically incorporated into a vehicle model. We want the reference trajectory to be feasible under the action of the actual controller, not only feasible dynamically. One aim of this work is to demonstrate that we can use complex control models that reflect the actual low-level controller in the planning stage.

## 6.2 Nested MPC for Integrated Planning and Control

MPC is a modern, flexible feedback control strategy that has shown promise for the control of autonomous vehicles. The flexibility of MPC has encouraged some authors to decompose the planning and control loops directly into a two-level MPC problem [83, 84, 85, 82]. Our approach builds on this strategy. The nested MPC approach for planning and control integration proposed here is simply an extension of MPC to control systems with multiple feedback loops of differing frequencies where the outer loop contains a derived model of the entire inner loop. The general schematic of this architecture is shown in Fig. 6.1 where the input to the local planner comes from some global plan. In our nested MPC design the local planner contains a model to predict the controlled plant just as the low-level controller contains a model of the plant. The robustness of the method comes from the feedback of both loops. Provided the frequencies are appropriately chosen model mismatch, controlled

model deficiencies and environmental uncertainties can be handled.

Numerous methods have been developed to find collision-free paths given arbitrary environments with non-static, non-convex obstacles. Many of these were developed in robotics and have significant research behind them, *e.g.* sampling methods like RRT [87]. The contribution of our work is not in refining this search for collision-free paths but to take such a path and find a parameterization that guarantees the low-level controller can track it within a specified tolerance. Our method is especially suited for combining with sample-based path generation methods which require a 'smoothing' step. In our proposed design we decompose the local path planner into two parts, see Fig. 6.2. The first part is a collision-free path generator that can be chosen amongst a plethora of options. In future work we would like to increase the level of integration by tailoring our method to particular path generation algorithms. The work presented here is concerned with the second component, a path parameterization method (or path 'smoothing' method). The goal of the path parameterizer is to determine a controller feasible path parameterization given some pre-determined collision-free path. In Fig. 6.3, the vehicle at 2 different timesteps is given by the hashed rectangle and the reference trajectory waypoints determined by the path parameterization procedure are given by the solid circles. The path parameterizer uses the nested MPC strategy by incorporating a model of the controlled vehicle. In our design the nesting is genuine, in that, just as the MPC for the vehicle controller incorporates a model of the plant dynamics for prediction, the planner uses a model of the controlled plant that is derived directly from the vehicle control MPC. Further, our proposed scheme can be written entirely explicitly so that derivative-based optimizers can be applied to the problem, yielding fast turnaround times.

### 6.2.1 Vehicle Control MPC

We use a direct transcription formulation of our MPC with control parameterization [61]. This minimizes the size of our optimization problem, which is advantageous for both com-

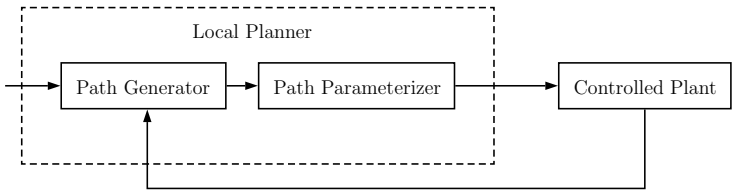



Figure 6.2: Local planner decomposition.

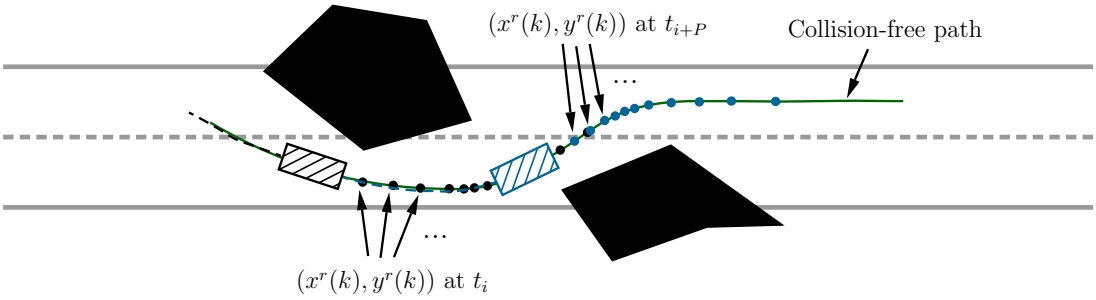



Figure 6.3: Illustration of path parameterization.

putational speed and our nested MPC design.

At each timestep we wish to solve the following NLP

$$\mathbf{U}^* = \underset{\mathbf{U}\in\mathcal{U}}{\arg\min}\ J_1(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r) \tag{6.1}$$

where $\mathbf{U}$ is the control sequence over the prediction horizon comprised of $H$ timesteps, $\mathcal{U}$ captures the control input limits, $\mathbf{x}(0)$ is the most recent measurement (or estimate) of the vehicle's state and $\mathbf{X}^r = [\mathbf{x}^r(1), \ldots, \mathbf{x}^r(H)]$ is the reference trajectory over the horizon. In our control design the reference trajectory only specifies the positions and velocities, no other states, *i.e.* $\mathbf{x}^r = \begin{pmatrix} x^r & y^r & v^r & 0 & 0 & 0 & 0 \end{pmatrix}^T$. We select the cost to be

$$J_1(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r) = \sum_{k=1}^{H} \|\tilde{\mathbf{x}}(k) - \mathbf{x}^r(k)\|^2_{\mathbf{Q}_1} + \|\mathbf{u}(k-1)\|^2_{\mathbf{R}_1} + \sum_{k=1}^{H-1} \|\Delta\mathbf{u}(k)\|^2_{\Delta\mathbf{R}_1}$$

where $\tilde{\mathbf{x}}(k)$ are the predicted states defined as functions of $\mathbf{x}(0), \mathbf{U}$ using the symSS approach. The discrete time model $\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k); \Delta t)$ is defined by the explicit Euler method given by

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \Delta t \Phi(\mathbf{x}(k), \mathbf{u}(k)), \tag{6.2}$$

applied to $\Phi$ the vector field of the nominal vehicle model given in Appendix A. The other components are defined as $\Delta\mathbf{u}(k) = (\mathbf{u}(k) - \mathbf{u}(k-1))/\Delta t$ and $\mathbf{Q}_1, \mathbf{R}_1, \Delta\mathbf{R}_1$ are tunable diagonal weighting matrices. This choice of cost function balances two objectives: to minimize the deviation from the reference trajectory and to minimize the inputs and their rates of action.

Further, in the interest of simplifying our problem we can take advantage of the fact $\mathcal{U}$ is well-behaved (*i.e.* compact and convex) to turn NLP (6.1) into an unconstrained NLP. We do this by augmenting the cost $J_1$ with a suitable penalty function $P_1$. The new unconstrained NLP is

$$\mathbf{U}^* = \arg\min_{\mathbf{U}} \; J_1(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r) + P_1(\mathbf{U}) \tag{6.3}$$

where

$$P_1(\mathbf{U}) = \sum_{k=1}^{H} \|\rho_p(\mathbf{u}(k-1))\|^2_{\mathbf{P}_1},$$

and the half-penalty function $\rho_p$ is the penalty function given in Section 3.1. $\mathbf{P}_1$ is another tunable weighting matrix.

NLP (6.3) has been purposefully formulated so that simple and fast optimization methods can be used. Further, this formulation is necessary for us to construct our nested MPC. The controller operates by solving NLP (6.3) and then implementing only $\mathbf{u}^*(0)$ at each timestep. We denote the function that returns $\mathbf{U}^*$ having solved NLP (6.3) by $\Omega$, *i.e.* $\Omega(\mathbf{x}(0), \mathbf{X}^r) = \mathbf{U}^*$.

### 6.2.2 Controlled Plant Model

In an MPC we approximate the dynamics of a plant modelled by differential equations using one-step methods since typically no closed form solution describing the dynamics exists. For our nested MPC design we require a model of the controlled plant. This presents a unique challenge since the controller solves an NLP at each timestep and so finding an approximation of its future actions is difficult.

If our control NLP is a convex unconstrained problem then a simple method like gradient descent, although slow, is guaranteed to converge to the optimum. Thus we can approximate the controller $\Omega(\mathbf{x}(0), \mathbf{X}^r)$ by $g^N(\mathbf{U}_0; \mathbf{x}(0), \mathbf{X}^r)$ for large $N \in \mathbb{N}$ with initial guess $\mathbf{U}_0$ where

$$g(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r) = \mathbf{U} - \gamma \nabla L_1(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r)$$

implements a single gradient descent step. The parameter $\gamma$ controls the descent rate and $L_1 = J_1 + P_1$ is the objective function of the vehicle controller NLP (6.3).

To improve this approximation further and cut down on computation cost we can augment the gradient descent step using

$$g'(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r) = \mathbf{U} - \gamma \mathbf{D}^{-1} \nabla L_1(\mathbf{U}; \mathbf{x}(0), \mathbf{X}^r) \tag{6.4}$$

where $\mathbf{D}$ is a diagonal matrix whose entries are the diagonal elements of the Hessian of $L_1$. This augmentation is particularly useful for control parameterized MPC since the timestep often acts as a small parameter that makes the Hessian of $L_1$ diagonally dominant [46]. Further improvement computationally could be made by making use of a dimensional reduction procedure [45]. In our implementation we set $N = 1$ and used the diagonals of the exact Hessian computed at the most recent iteration of the low-level control MPC.

### 6.2.3 Path Parameterizing MPC

The path parameterizer is the second stage of the local path planner and provides the step to integrate the planning and control routines. It relies on an upstream path generation routine that can deliver a collision-free path with some error tolerance. We suppose the collision free path is given by a spline $\mathbf{s}(\tau) = (s_x(\tau), s_y(\tau)), \tau \in [0, 1]$ in the global coordinate frame. In this work we assume the splines are cubic Hermite splines (which are equivalent to cubic Bézier splines). The associated reference trajectory components are then determined by $(x^r(k), y^r(k), v^r(k)) = (s_x(\tau(k)), s_y(\tau(k)), \upsilon(k))$, where $\upsilon(k) = \|\mathbf{s}(\tau(k+1)) - \mathbf{s}(\tau(k))\|/\Delta t$ meaning that $\mathbf{X}^r$ is determined by $\mathbf{s}$ and the $\tau(k)$s. Thus given the spline from the path generation phase, the planner then solves for the reference positions via $\tau(k) \in [0, 1]$ and the associated velocities such that $\upsilon(k) \in [v_{\min}, v_{\max}]$. An advantage of this setup is that the optimization variables belong to compact and convex sets making the solution process simpler.

At a planning step, which occurs once every $P$ timesteps, we wish to solve the NLP

$$\mathbf{T}^* = \underset{\mathbf{T}\in\mathcal{T}}{\arg\min}\ J_2(\mathbf{T}; \mathbf{x}(0), \mathbf{s}) \tag{6.5}$$

where $\mathbf{T} = [\tau(1), \ldots, \tau(M)]$, $M = S + H - 1, S \geq P$ is the number of steps we simulate the controlled plant and $\mathcal{T}$ captures the bounds on the optimization variables. Since $\mathbf{s}$ is assumed known and fixed we can use $\mathbf{T}$ and $\mathbf{X}^r$ interchangeably. From an initial condition of $\mathbf{x}(k), \mathbf{U}_{k-1}$ with reference trajectory determined by $\mathbf{T}$ we can simulate the controlled plant one step forward using Eqn. (6.2) where $\mathbf{u}(k)$ is the first entry of $\mathbf{U}_k$, which is the solution to

$$\mathbf{U}_k = g'^N(\mathbf{U}_{k-1}; \mathbf{x}(k), \mathbf{T})$$

where $g'$ is as defined in Eqn. (6.4). Further, to increase the effective planning horizon length, we simulate the plant over the final controller prediction horizon by recursively evaluating Eqn. (6.2) beginning with $\mathbf{x}_S$ where the control inputs are $[\mathbf{u}(S), \ldots, \mathbf{u}(S + H - 1)] = \mathbf{U}_S$.

The cost is chosen to be

$$J_2(\mathbf{T};\mathbf{x}(0),\mathbf{s}) = \sum_{k=1}^{S+H-1} \|\Delta\mathbf{x}(k)\|^2_{\mathbf{Q}_2} + \sum_{k=1}^{S} \|\rho_p(\Delta\mathbf{x}(k))\|^2_{\mathbf{Q}'_2}$$

where $\Delta\mathbf{x}(k) = \begin{pmatrix} \|\Delta(x,y)\|^2 \\ |\Delta v|^2 \end{pmatrix}$ is the difference in trajectory from the reference is a function of $\mathbf{T}, \mathbf{x}(0), \mathbf{s}$ with $\Delta(x,y) = (x(k), y(k)) - \mathbf{s}(\tau(k))$ and $\Delta v = v(k) - \upsilon(k)$ and $\mathbf{Q}_2, \mathbf{Q}'_2$ are weighting matrices. The first term aims to minimize the difference in trajectory tracking over the simulated controlled plant steps and the final horizon steps. The second term is to enforce the error tolerance of our collision free region $\|\Delta(x,y)\| < \delta_{(x,y)}$ and $|\Delta v| < \delta_v$. Similar to the vehicle control MPC we include a penalty function to enforce the bounds on the optimization variables via

$$P_2(\mathbf{T};\mathbf{s}) = \sum_{k=1}^{S+H-1} \|\rho_p(\mathbf{v}(k))\|^2_{\mathbf{P}_2}$$

where $\mathbf{v}(k) = \begin{pmatrix} \tau(k) & \upsilon(k) \end{pmatrix}^T$ and $\mathbf{P}_2$ is a weighting matrix, to arrive at the unconstrained NLP

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \; J_2(\mathbf{T};\mathbf{x}(0),\mathbf{s}) + P_2(\mathbf{T};\mathbf{s}). \tag{6.6}$$

The path parameterization MPC runs by solving NLP (6.6) every $P$ timesteps to determine the reference trajectories to be fed to the low-level MPC controller until the next planning step. At the $i^{th}$ step after the planning step the reference trajectory fed to the low-level controller is determined by $[\tau(i+1), \ldots, \tau(i+H+1)]$. In this work we used gradient descent to solve NLP (6.6) since we could compute the Hessian of $L_2$, but it proved to be too large for MATLAB to convert to a MEX function on our workstation. We outline how NLP (6.6) can be solved using derivative based optimization by taking advantage of symbolic computing in the next section.

## 6.3 Case Study

### 6.3.1 Implementation Details

To achieve fast solutions we make use of automatic code generation and optimization to produce routines to evaluate the two objective functions $L_1, L_2$ and their exact derivatives.

Table 6.1: Nested MPC parameters.

| Parameter | Value |
|---|---|
| $H, P, S$ | 10, 5, 7 |
| $\Delta t$ | 0.05 (s) |
| $\mathbf{Q}_1$ | $\text{diag}\begin{pmatrix}2\times10^4 & 2\times10^4 & 100 & 0 & 0 & 0 & 0\end{pmatrix}$ |
| $\mathbf{R}_1$ | $\text{diag}\begin{pmatrix}100 & 0.01 & 500\end{pmatrix}$ |
| $\Delta\mathbf{R}_1$ | $\text{diag}\begin{pmatrix}100 & 0.001 & 5000\end{pmatrix}$ |
| $\mathbf{P}_1$ | $\text{diag}\begin{pmatrix}100 & 1000 & 1000\end{pmatrix}$ |
| $\mathbf{Q}_2$ | $\text{diag}\begin{pmatrix}10^4 & 100\end{pmatrix}$ |
| $\mathbf{Q}_2'$ | $\text{diag}\begin{pmatrix}100 & 0\end{pmatrix}$ |
| $\mathbf{P}_2$ | $\text{diag}\begin{pmatrix}10^4 & 0\end{pmatrix}$ |
| $p$ | 4 |
| $\delta_{(x,y)}, \delta_v$ | 0.1 (m), 0.5 (m/s) |
| $\gamma$ | 2 |
| $v_{\min}, v_{\max}$ | 10 (km/h), 100 (km/h) |

We used Maple to do this fast, flexibly and efficiently. Using the symSS formulation for the vehicle MPC the controlled plant model is then in an explicit form so we can apply the symSS approach to the path parameterizer in a straightforward manner.

The objective functions and their derivatives were generated and optimized as standalone functions. These functions were then converted to MEX functions that we then used in our NLP solvers. All simulations were run using MATLAB 2017b on a desktop with an Intel(R) Core(TM) i7-4790 CPU. The parameters used in the Results section are given in Tab. 6.1.

### 6.3.2 Results

To demonstrate the role the path parameterizer can play in an integrated path planner and controller, we first fix the collision free path and simulate the plant using the same vehicle model used in the MPCs (see: Appendix A). In the first scenario it is given by a spline tracing out a zig-zag pattern. Figure 6.4 displays some sample simulations comparing 0 parameterization steps (*i.e.* gradient descent steps applied to NLP (6.6)) and 100 or 200 parameterization steps. At each planning step the initial guess $\mathbf{T}_0$ matches a

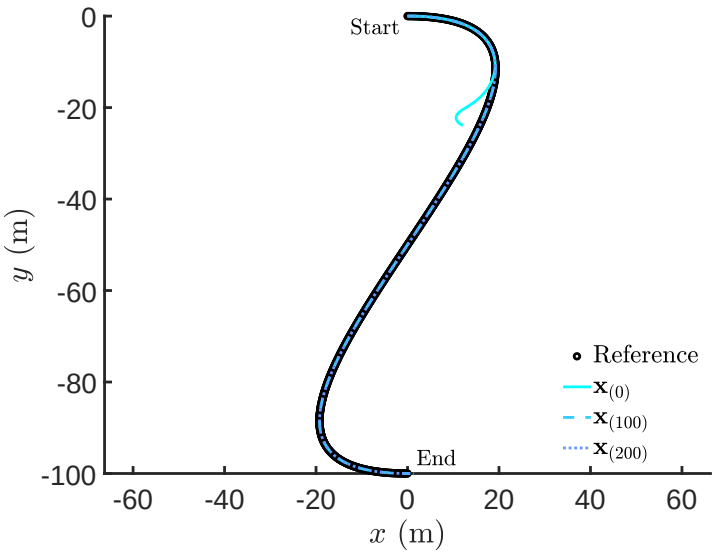



Figure 6.4: Position tracking results for zig-zag maneuver.

constant velocity trajectory along the spline at 27 km/h. As can be seen from Fig. 6.4 the simulation with no path parameterization fails to follow the reference after the first turn. In Fig. 6.5 the reference tracking errors are presented. The maximum position error is 8.06 cm and 5.44 cm for the 100 and 200 step simulations, respectively. Similarly, the maximum velocity errors are 37.10 cm/s and 28.92 cm/s, respectively. Thus, we can see that simulations with greater path parameterization performs better in terms of tracking error. Both solutions maintain their tracking errors below the chosen thresholds (shown by the dashed black lines). Lastly, in Fig. 6.6 the associated control signals are displayed. From this we can see that the simulations with path parameterization have reasonably smooth controls with small fluctuations about a trend. Furthermore, the mean computation time of the low-level control MPC is $4.6 \times 10^{-4}$s and the mean computation time of the path parameterizer is $7.6 \times 10^{-3}$s and $1.38 \times 10^{-2}$s for the 100 and 200 step simulations, respectively. Thus, as expected the computation time grows directly in proportion to the number of gradient descent steps. These computation time results mean that the controller and path parameterizer are faster than real-time on our architecture.

In Fig. 6.7 we can see the results of simulating a lane change maneuver. In this scenario the default speed is a constant 80 km/h. One notable aspect of the results is that those solutions with parameterization are able to almost eliminate cross-track error. The more parameterization steps present, the greater the damping of vehicle acceleration. Figure

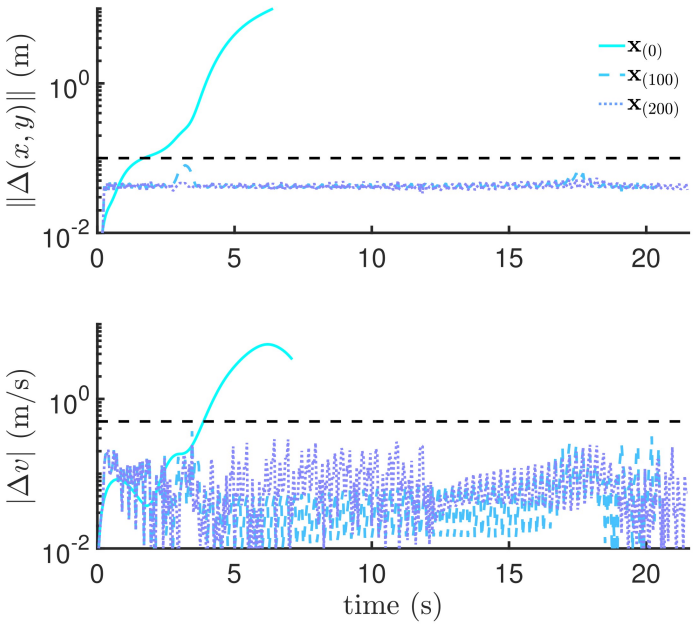



Figure 6.5: Error in trajectory tracking during zig-zag maneuver.

6.8 displays the associated errors. At this higher speed, more parameterization steps are required to meet the position and velocity error tolerances. The maximum position error for the 1000 step solution is 10.27 cm, only just greater than the 10 cm tolerance. The 500 step solution however performed better in terms of velocity tracking with a maximum error of 57.31 cm/s versus the 1000 step solution with a maximum of 62.98 cm/s. This observed violation of the constraints is to be expected since the constraints are only enforced by a soft penalty function. It is important to note that all these results are sensitive to the tuning parameters. We have weighted position tracking more heavily than velocity tracking, which is evident in these results.

Lastly, we integrated our path parameterization method with a rudimentary path generation scheme that used a simple sample and trim strategy. The velocity during the simulation is about $50 \pm 10$ km/h and simulates a vehicle travelling down a curvy two lane road with static obstacles. In Fig. 6.9 we display a snapshot of this simulation that shows the current vehicle position (black rectangle), its past positions (red circles), predicted future positions (magenta circles), the reference trajectory (cyan circles), the reference collision-free spline (cyan), other sampled collision-free paths (green), sampled non-collision-free paths (blue), and obstacle (red rectangle). To the right in Fig. 6.9 are the control input

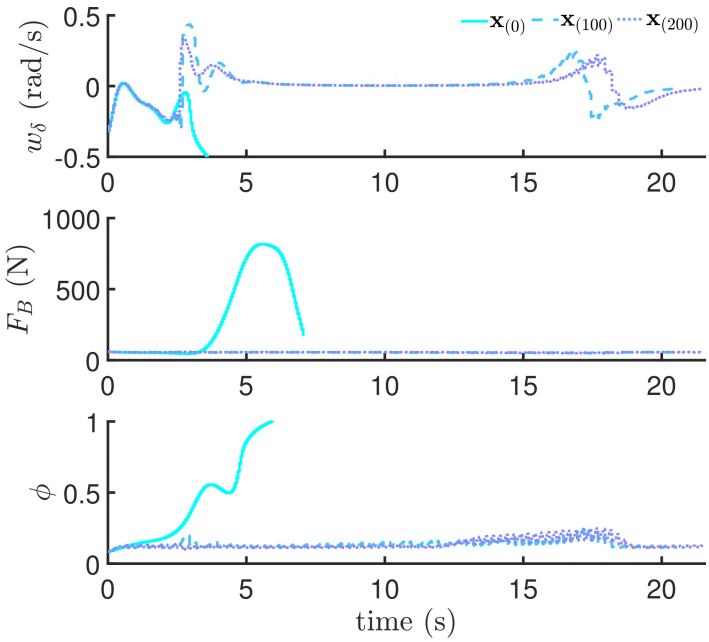



Figure 6.6: Control inputs during zig-zag maneuver.

histories (red) and future predictions (blue dashed) over the timesteps of the scenario. In Fig. 6.10 we display the tracking error for this short simulation. As we can see, the presence of obstacles and path resampling does not pose a problem for the path parameterizer. The path parameterizer is still able to provide a reference trajectory that the vehicle can follow within the allowed tolerance.

## 6.4 Discussion

We have introduced a method that can be integrated with existing path generation techniques to produce collision-free, controller feasible reference paths. This is a step towards solving the safety-critical problem of planning and control integration for autonomous vehicles. A feedback-based nested MPC design was proposed that can be implemented using derivative-based optimizers for fast turnaround times. The key to this design is the symSS formulation of the low-level control MPC (see: Chapter 3). From this base we could approximate its action by an explicit function and so incorporate it as a model within an outer

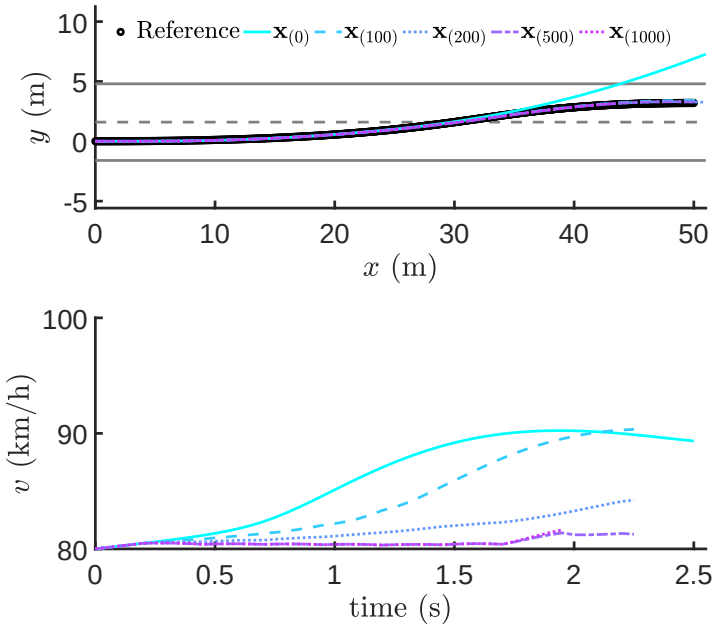



Figure 6.7: Lane change maneuver.

MPC called the path parameterization MPC. The results presented here are preliminary but hint at the usefulness of the method.

In future work, improvements to the NLP optimizers, as well as objective function definition and tunings, will be investigated for a wide range of realistic driving scenarios. In particular, the inclusion of yaw angle tracking will be included for better collision avoidance. We also wish to construct a complete integrated planning and control package by integrating our nested MPC with a robust and fast collision-free path generator. Future work should examine the utility of this level of integration in comparison to standard approaches. Particular care should be focused on special cases concerning difficult obstacle arrangements for which pure replanning is not satisfactory. Questions of stability and robustness of the nested MPC design have not been examined.

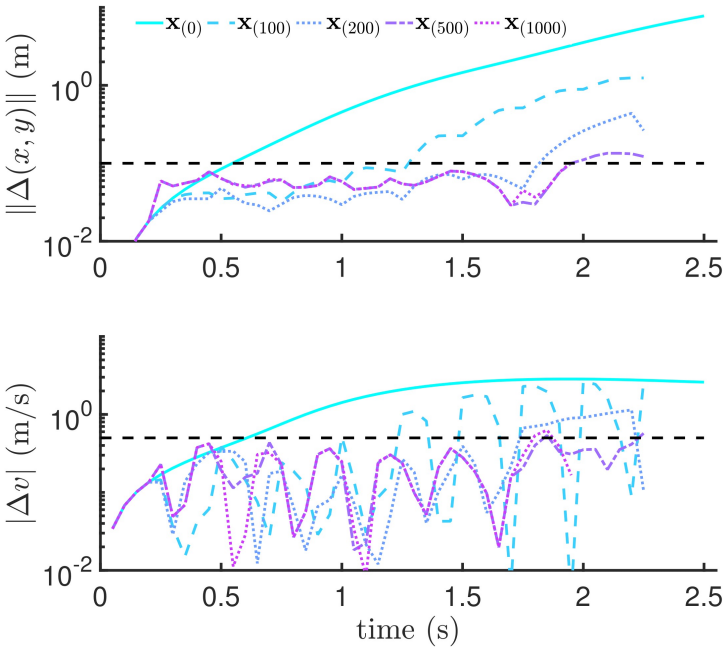



Figure 6.8: Lane change tracking error.

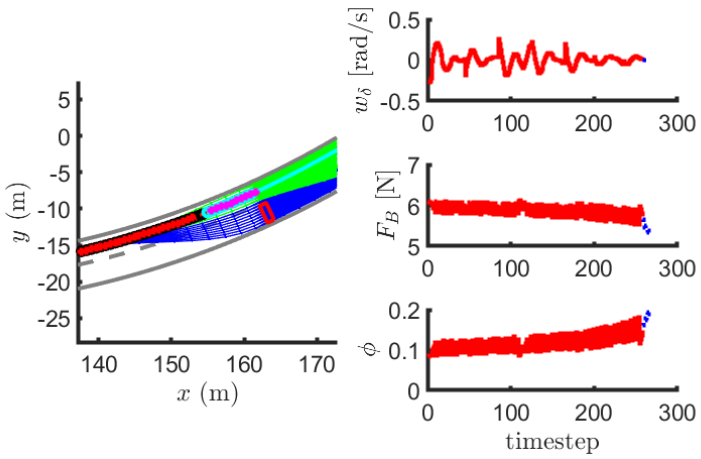



Figure 6.9: Snapshot of an obstacle avoidance simulation.

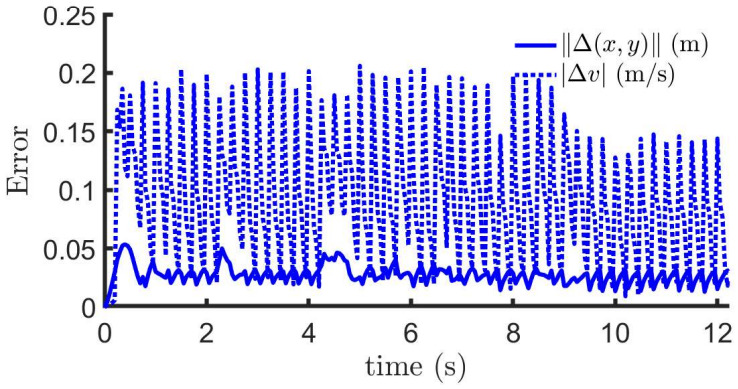



Figure 6.10: Reference tracking error during obstacle avoidance simulation.

# Chapter 7

# Fast Hybrid NMPC Using Quasi-Translations

In this chapter we present the quasi-translation (QT) strategy for mixed-integer MPC problems. This strategy handles the complexity introduced by discrete controls in a simple manner that takes advantage of the sequential nature of the FHOCPs. We restrict the set of possible discrete control sequences considered at each time step by using parameterized quasi-translations, which relate discrete controls over consecutive timesteps. The parameters can be tuned to balance controller performance, chatter and turnaround time. We also introduce an approach to handle hybrid models with state-dependent switches for use with a symbolic formulation of the MPC problem that integrates easily with the QT strategy.

We investigate the QT strategy for a variety of hybrid nonlinear MPC problems including the classic benchmark Lotka-Volterra fishing problem, an electric submarine problem, a diesel airpath control problem and an autonomous vehicle problem. Results show the QT strategy consistently yields a reduction in turnaround times with excellent or minimally degraded performance as compared to competing strategies. In some cases, the QT strategy is able to find desirable solutions that other methods could not. An investigation of the parameters of the method is also conducted indicating that the computational bound of the QT strategy can be kept quite small in practice.

## 7.1 Hybrid MPC Review

Discrete inputs, ranging from simple on/off buttons to complex commands expressed in a natural language, are ubiquitous among systems in our world. These inputs share the common feature that they belong to countable sets, *i.e.* there is a bijective mapping between them and a subset of the integers, and thus they are commonly deemed "integer controls". Being able to precisely control such systems presents a challenge since the lack of smooth inputs contradicts the underlying assumption of continuity present in optimal control theory.

Traditionally, the tools of control theory have been developed to handle systems governed by smooth dynamics and with smooth inputs. In recent years significant work has been devoted to the optimal control of hybrid systems that exhibit both smooth and discrete behaviours; see [88] and the references therein. Examples of hybrid systems include a human walking [89] and numerous automotive subsystems [90, 91]. These systems combine regimes with smooth dynamics along with a discrete switching logic. Controlling hybrid systems in an optimal fashion presents both a theoretical and practical challenge as such systems are ubiquitous. In the field of MPC, hybrid systems control has also been a recent focus of research, though much of the theoretical results are limited, as they pertain only to piecewise affine systems [92]. Problems with hybrid behaviour are still very challenging for real-time NMPCs.

Difficulties arise in solving (2.2) when discrete components are present in the problem. Two such situations that we address in this paper are: 1. when some components of the control $u$ belong to a countable set (*i.e.* are discrete-valued) or, 2. when the model $\phi$ has state-dependent switches. We note these situations are not mutually exclusive. For situation 1, we can reformulate the controls such that $\mathbf{u} \in \mathbb{R}^{m-d} \times \mathbb{Z}^d$ with $d \geq 1$.

In this chapter we extend the QT strategy to the second case of fast hybrid system control using NMPC. Both situations in fact fall under the umbrella of hybrid systems since discrete behaviour is present in the models of both. In the optimal control literature, hybrid systems have been classified as either externally forced systems (EFS), those with discrete inputs (situation 1), or internally forced systems (IFS), those with state dependent switching (situation 2) [88, 93]. An EFS is said to contain explicit (or controllable) switches while an IFS has implicit (or state-dependent) switches. In the optimal control literature, IFS systems are considered more difficult to handle but recent methods have been proposed to mitigate some of the issues [94]. In the MPC setting the problem is simplified and we propose a fast strategy to handle the implicit switching behaviour of these systems efficiently.

For externally forced systems it is straightforward to cast Problem (2.2) as a Mixed-Integer Nonlinear Programming (MINLP) problem. However, the complexity that the discrete controls introduce remains and must be handled by a MINLP solver and it is known that MINLPs are NP-Hard and can even be undecidable [15, 16, 95]. There are a multitude of MINLP solver methods, see [96, 97].These solvers typically operate by reformulating and decomposing the MINLP into simpler subproblems like MILPs or NLPs. For example, techniques such as Branch and Bound (BB) and Cutting Plane methods perform a tree search of the integer controls space in order to decrease the computational cost [98, 99, 75]. However these solvers can suffer from an exponential growth in solution complexity if all branches of the tree need to be visited. Further, they often rely on heuristics, which greatly affects the success of the method. These properties make them unsuitable for real-time MPC applications The QT strategy, presented here, provides a method with a hard upper bound on the solution complexity in terms of design parameters. Essentially, this strategy uses realistic assumptions about the discrete control inputs to dramatically reduce the computation.

## 7.2 Quasi-Translations for Externally Forced Hybrid NMPC

The development of QTs for externally forced hybrid NMPC is guided by two assumptions:

1. our future predictions are *almost* perfect, and
2. the present is inevitable.

We translate these assumptions into constraints on integer controls via the following argument. Suppose we wish to control a system with a single binary input using NMPC. At any given timestep our NMPC returns an optimal binary input sequence of length $H$ (the size of the horizon) of which we implement only the first element. If our future predictions were perfect, then exactly that sequence of inputs will be implemented over the next $H$ steps. In particular, any switches in control input will be realized as predicted $H$ timesteps previously. So from one timestep to the next we would see any switches predicted in the horizon advance one step closer to being realized. Since our future predictions are not exactly perfect, the timing of the switches in the solution from one timestep to the next should not differ by more than some upper bound. This constraint implements the first assumption. Our second assumption tells us that new switches in the control sequence can

only be introduced at the end of the horizon, *i.e.* they can only be realized after being predicted.

The effect of these assumptions is twofold. First, we can dramatically reduce the set of integer control sequences under consideration and thus bound the computational cost of the NMPC. For a binary control there are at worst $2^H$ binary control sequences to consider. Many MINLP solvers employ a tree search strategy that may be able to find an optimum by enumerating many fewer cases but in the worst case they may still require a full enumeration. Secondly, we can control the chatter of the integer controls to restrict undesirable behaviour. Loosely speaking, a QT of an integer control sequence is any other integer control sequence that shares the same approximate switch timings and for which any switches not present in the original are near the end of the sequence.

During the online operation of the MPC, the set of QTs of the integer controls of the solution at the previous timestep are used as candidates. The task of choosing an initial integer control sequence is not addressed here. One possibility is to utilize precomputed MINLP solutions for a variety of realistic initial conditions as potential warm-start values.

### 7.2.1 Background

For purposes of exposition we consider purely binary controls in the following. This is not as restrictive as it may first seem since all integer controls can be rewritten using a combination of binary ones [100].

Let us denote a binary sequence by $\mathbf{B} = [\mathbf{b}(1), \ldots, \mathbf{b}(H)] \in \{0,1\}^H$. A *switch* occurs when consecutive elements differ. The number of switches $s$ a binary sequence has is $s(\mathbf{B}) = \sum_{k=1}^{H-1} |\mathbf{b}(k+1) - \mathbf{b}(k)|$. A consecutive subsequence of elements denoted by $\mathbf{B}_{[i,i+N]} = [\mathbf{b}(i), \mathbf{b}(i+1), \ldots, \mathbf{b}(i+N)]$ such that

$$[s(\mathbf{B}_{[i,i+N]}) = 0] \wedge [i > 1 \Rightarrow s(\mathbf{B}_{[i-1,i+N]}) = 1] \wedge [i + N < H \Rightarrow s(\mathbf{B}_{[i-1,i+N+1]}) = 1]$$

is called a *block*, *i.e.* a consecutive subsequence of equal elements of maximal length. When both $i > 1$ and $i + N < H$ the block is called an *internal block*. We call $\mathbf{b}(i)$ the *front*, $\mathbf{b}(i+N)$ the *end* and $N+1$ the *length* of the block. Thus, the number of blocks in a sequence $\mathbf{B}$ is $s(\mathbf{B}) + 1$ and the number of internal blocks is $\max\{s(\mathbf{B}) - 1, 0\}$. We denote the set of block fronts by $F(\mathbf{B}) = \{i \in \{2, \ldots, H\} | \mathbf{b}(i) \neq \mathbf{b}(i-1)\}$ and the set of block ends by $E(\mathbf{B}) = \{i \in \{1, \ldots, H-1\} | \mathbf{b}(i) \neq \mathbf{b}(i+1)\}$. So $s(\mathbf{B}) = |F(\mathbf{B})| = |E(\mathbf{B})|$. Further, the length of the smallest internal block is given by

$$l(\mathbf{B}) = \begin{cases} 0 & \text{if } |E(\mathbf{B})| \leq 1 \\ \min\{e(i+1) - e(i) | e(i) \in E(\mathbf{B})\} & \text{otherwise} \end{cases}.$$

As an example, consider $\mathbf{B} = [1, 1, 0, 0, 1, 1, 1]$ then

$$s(\mathbf{B}) = 2, E(\mathbf{B}) = \{2, 4\}, F(\mathbf{B}) = \{3, 5\}, l(\mathbf{B}) = 2.$$

The above definitions can be straightforwardly generalized to arbitrary integer sequences.

To control the switching frequency of our controller we consider only those sequences with a maximum number of switches $0 \leq s \leq H - 1$ and a minimum internal block length $1 \leq l \leq H - 2$. Using these parameters we can define the set of sequences with a maximum number of switches $s$ and minimum internal block length $l$,

$$\mathcal{B}(s, l) = \{\mathbf{B} \in \{0, 1\}^H | [s(\mathbf{B}) \leq s] \wedge [l(\mathbf{B}) \geq l]\}.$$

Determining the cardinality of $\mathcal{B}$ is a difficult problem belonging to the field of restricted compositions in combinatorics. No straightforward closed formula exists for an arbitrary choice of parameters. Using well-known results for integer compositions we can provide a simple bound in terms of $s$, $|\mathcal{B}(s, l)| \leq |\mathcal{B}(s, 1)| = \binom{H-1}{s-1}$.

Next we introduce two types of QTs. There are potentially many others to be explored but we limit our discussion to the following.

### 7.2.2 Crab-Walk Quasi-Translations

In this strategy the set of allowable sequences at a given timestep are constrained to be stiff translations of the solution at the previous timestep. Thus the length of the internal blocks are fixed and hence the blocks move along the prediction horizon over multiple timesteps akin to a crab walking.

We define the set of Crab-Walk QTs (CWQTs) of a sequence $\mathbf{B}$ by

$$\mathcal{R}(\mathbf{B}; s, l, r) = \mathbf{B} \cup \left(\mathcal{R}^f(\mathbf{B}; r) \cup \mathcal{R}^{\mathbf{b}}(\mathbf{B}; r)\right) \cap \mathcal{B}(s, l)$$

where $r \geq 1$ is an upper limit on the number of steps taken,

$$\mathcal{R}^f(\mathbf{B}; r) = \{\mathbf{B}' \in \{0, 1\}^H | [\mathbf{B}'_{[1,H-r']} = \mathbf{B}_{[r'+1,H]}] \wedge [r' \leq r]\}$$

is the set of allowed forward translations and

$$\mathcal{R}^{\mathbf{b}}(\mathbf{B}; r) = \{\mathbf{B}' \in \{0, 1\}^H | [b'(1) = \ldots = b'(r') = \mathbf{b}(1)] \wedge [\mathbf{B}'_{[r'+1,H]} = \mathbf{B}_{[1,H-r']}] \wedge [r' \leq r]\}$$

is the set of allowed backward translations. Note that in $\mathcal{R}^b$ we restrict elements at the beginning of the sequence so no new switches occur, unlike $\mathcal{R}^f$ where we allow new elements in the tail to be arbitrary. This is done to mitigate chatter since any new switches

introduced at the end will have to be translated to the front of the sequence over multiple timesteps and cannot occur arbitrarily at the front. As an example, if $\mathbf{B} = [1, 1, 0, 0, 1, 1, 1]$ then

$$\begin{aligned}\mathcal{R}(\mathbf{B}; 2, 1, 2) = \{&[0, 0, 1, 1, 1, 0, 0],\\ &[0, 0, 1, 1, 1, 1, 0],\\ &[0, 0, 1, 1, 1, 1, 1],\\ &[1, 0, 0, 1, 1, 1, 1],\\ &[\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}],\\ &[1, 1, 1, 0, 0, 1, 1],\\ &[1, 1, 1, 1, 0, 0, 1]\}.\end{aligned}$$

The parameters $s, l, r$ allow us to limit the size of $\mathcal{R}$. By a simple counting argument we can provide an upper bound that is independent of $H$

$$|\mathcal{R}(\mathbf{B}; s, l, r)| \leq |\mathcal{R}(\mathbf{B}; H - 1, 1, r)| \leq 2^{r+1} + r - 1.$$

### 7.2.3 Inchworm Quasi-Translations

In this strategy a subset of the ends or fronts of blocks have their values swapped so that internal blocks are allowed to grow and shrink. In this way the internal blocks can move along the horizon over multiple timesteps like an inchworm.

The set of Inchworm QTs (IQTs) of a sequence $\mathbf{B}$ is defined as

$$\mathcal{S}(\mathbf{B}; s, l, p) = \mathbf{B} \cup \left(\mathcal{S}^{f}(\mathbf{B}; p) \cup \mathcal{S}^{\mathbf{b}}(\mathbf{B}; p)\right) \cap \mathcal{B}(s, l)$$

where $p \geq 1$ is an upper limit on the number of elements that can be swapped,

$$\begin{aligned}\mathcal{S}^{f}(\mathbf{B}; p) = \{\mathbf{B}' \in \{0, 1\}^{H} | [b'(i) = \mathbf{b}(i + 1), i \in D] \wedge [b'(i) = \mathbf{b}(i), i \notin D \cup \{H\}],\\ D \subseteq E(\mathbf{B}), 0 \leq |D| \leq p\} \setminus \mathbf{B}\end{aligned}$$

is the set of allowed forward translations and

$$\begin{aligned}\mathcal{S}^{\mathbf{b}}(\mathbf{B}; p) = \{\mathbf{B}' \in \{0, 1\}^{H} | [b'(i) = \mathbf{b}(i - 1), i \in D] \wedge [b'(i) = \mathbf{b}(i), i \notin D],\\ D \subseteq F(\mathbf{B}), 1 \leq |D| \leq p\}\end{aligned}$$

is the set of allowed backwards translations. Similar to the crab-walk strategy only the forward set allows new switches to be introduced and so switches must propagate over timesteps from the tail to the beginning of the sequence.As an example, if $\mathbf{B} = [1, 1, 0, 0, 1, 1, 1]$ then

$$\begin{aligned}\mathcal{S}(\mathbf{B}; 2, 1, 2) = \{&[1, 0, 0, 0, 1, 1, 1],\\ &[1, 0, 0, 1, 1, 1, 1],\\ &[1, 1, 0, 1, 1, 1, 1],\\ &[\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}],\\ &[1, 1, 0, 0, 0, 1, 1],\\ &[1, 1, 1, 0, 0, 1, 1],\\ &[1, 1, 1, 0, 1, 1, 1]\}.\end{aligned}$$

Similar to the crab-walk strategy we can provide an upper bound on the size of $\mathcal{S}$ that is independent of $H$

$$|\mathcal{S}(\mathbf{B}; s, l, p)| \leq |\mathcal{S}(\mathbf{B}; s, 1, s)| \leq 3 \times 2^s - 2.$$

Both strategies realize our guiding assumptions by careful selection of forward and backward sets. The parameters allow us to restrict the size of the set of possibilities and to control the chatter frequency by limiting the rate of translation and time between switches.

## 7.3 Greedy Quasi-Translation Optimization for NMPC

Using QT strategies we can restrict the potential integer control sequences to a much smaller set of possibilities at each timestep of an MPC. However, a full enumeration approach of this smaller set may still be prohibitive since the size of the set of QTs scales exponentially with the design parameters, $s$ and $r$ or $p$, in the worst case. Following the intuition of our first assumption further, we present a greedy search method to find an approximate optimal QT. The greedy aspect of the search method results in some candidate solutions being missed. The algorithms scale logarithmically with the size of the set of potential QTs, which drastically cuts down on the computation time. Further, if there are multiple integer controls, these search algorithms can be interleaved so that the additional cost is only additive.

In the following we present an algorithm for each of the QTs introduced. Both of these utilize a greedy scheme with a recursive sub-routine that simultaneously optimizes

the discrete and continuous inputs. The search algorithms, Algorithm 2 and Algorithm 3, proceed by looking for decreases in the objective function by making sequential single-element QTs beginning at the first element and then progressing to the end of the horizon. This is done in tandem with the simultaneous optimization of the continuous variables using a NLP routine. Once an improvement is found, the algorithms proceed within the forward or backward set in which the improvement was first found. The greedy CWQT algorithm acts by sliding internal blocks forward or backwards and the greedy IQT algorithm acts by swapping the ends or fronts of blocks. Both are biased to look for improvements in the forward set first.

We denote the NLP routine that solves Problem (2.2) with fixed integer inputs $\mathbf{B}$ and returns the objective function value by $\Gamma(\mathbf{B})$. We define $\Gamma(\emptyset) = \infty$. Evaluating $\Gamma$ is by far the most computationally expensive sub-routine as it solves an NLP problem and thus its calls should be minimized. Algorithm 2 finds an approximate CWQT optimum with worst-case performance $(2r + 1) \times O(\Gamma)$ and Algorithm 3 finds an approximate IQT optimum and has worst-case performance $(2s + 1) \times O(\Gamma)$.

## 7.4 Internally Forced Hybrid Systems NMPC

Internally forced hybrid systems (or implicitly switched systems) are notoriously difficult to handle in an optimal control setting. In the following we describe a method to handle such systems by introducing artificial controls.

A discrete-time hybrid model with implicit switches can be described by the model subsystems (or modes) $f_m$ and switching functions $\sigma_m^i$, $m \in \mathcal{M}, i \in \{1, \ldots, m_n\}$. In the following we assume there are no state jumps (or discontinuities) at the mode transitions and the switching functions are reasonably smooth. In more detail we can describe such a system by

$$\mathbf{x}(k+1) = f_m(\mathbf{x}(k), \mathbf{u}(k); \Delta t)$$

where $m \in \mathcal{M}$ is the active mode. A mode $m$ is active when $(\mathbf{x}, \mathbf{u}) \in \mathcal{A}_m = \{(\mathbf{x}, \mathbf{u}) | \sigma_m^i(\mathbf{x}, \mathbf{u}) \leq 0, i = 1, \ldots, m_n\}$. We assume the system model is well-defined so that all $\mathcal{A}_m, m \in \mathcal{M}$ are disjoint and that the model domain is contained in $\bigcup_{m \in \mathcal{M}} \mathcal{A}_m$.

We can handle the difficulties presented by hybrid systems in the symSS formulation by introducing artificial controls $\mathbf{w}$ corresponding to the hybrid system's modes. We then embed the hybrid model into the larger class of continuous systems allowing us to generate exact derivatives despite the discrete behaviour of the system. The resulting computation

**Algorithm 2** Greedy Search for Optimal $\mathbf{B}' \in \mathcal{R}(\mathbf{B}; s, l, r)$

**function** GREEDYCRABWALK($\mathbf{B}$)
    $C \leftarrow \Gamma(\mathbf{B})$
    **return** STEP($\mathbf{B}, C,$ `forward`$, 0$)

**function** STEP($\mathbf{B}, C,$ `direction`, `count`)
    **if** `count` $\geq r$ **then**
        **return B**
    **if** `direction` is `forward` **then**
        $\mathbf{B}' \leftarrow \{[\mathbf{B}_{[2,H]}, \mathbf{b}(H)]\} \cap \mathcal{B}(s, l)$
        $\mathbf{B}'' \leftarrow \{[\mathbf{B}_{[2,H]}, \neg\mathbf{b}(H)]\} \cap \mathcal{B}(s, l)$
    **else**
        $\mathbf{B}' \leftarrow \{[\mathbf{b}(1), \mathbf{B}_{[1,H-1]}]\} \cap \mathcal{B}(s, l)$
        $\mathbf{B}'' \leftarrow \emptyset$
    $C' \leftarrow \Gamma(\mathbf{B}')$
    $C'' \leftarrow \Gamma(\mathbf{B}'')$
    **if** $\min\{C, C', C''\} = C$ **then**
        **if** `count` $= 0$ **then**
            **return** STEP($\mathbf{B}, C,$ `backward`, `count`)
        **else**
            **return B**
    **else if** $\min\{C, C', C''\} = C'$ **then**
        **return** STEP($\mathbf{B}', C',$ `direction`, `count` $+ 1$)
    **else**
        **return** STEP($\mathbf{B}'', C'',$ `direction`, `count` $+ 1$)

**Algorithm 3** Greedy Search for Optimal $\mathbf{B}' \in \mathcal{S}(\mathbf{B}; s, l, p)$

**function** GREEDYINCHWORM($\mathbf{B}$)
  $E \leftarrow E(\mathbf{B}) \cup \{H\}$
  $F \leftarrow F(\mathbf{B})$
  $C \leftarrow \Gamma(\mathbf{B})$
  **return** INCH($\mathbf{B}, C, E, F, 0$)

**function** INCH($\mathbf{B}, C, E, F,$ `count`)
  **if** $E = \{H\}$ and `count` $\geq p$ **then**
    `count` $\leftarrow$ `count` $- 1$
  **if** $E \cup F = \emptyset$ or `count` $\geq p$ **then**
    **return B**
  $m \leftarrow \min\{E \cup F\}$
  **if** $m = H$ **then**
    $\mathbf{B}' \leftarrow \{[\mathbf{B}_{[1,H-1]}, \neg\mathbf{b}(H)]\} \cap \mathcal{B}(s, l)$
  **else if** $m \in E$ **then**
    $\mathbf{B}' \leftarrow \{[\mathbf{B}_{[1,m-1]}, \mathbf{b}(m+1), \mathbf{B}_{[m+1,H]}]\} \cap \mathcal{B}(s, l)$
  **else**
    $\mathbf{B}' \leftarrow \{[\mathbf{B}_{[1,m-1]}, \mathbf{b}(m-1), \mathbf{B}_{[m+1,H]}]\} \cap \mathcal{B}(s, l)$
  $C' \leftarrow \Gamma(\mathbf{B}')$
  **if** $C \geq C'$ and $m \in E$ **then**
    **return** INCH($\mathbf{B}', C', E \setminus \{m\}, \emptyset,$ `count` $+ 1$)
  **else if** $C \geq C'$ and $m \in F$ **then**
    **return** INCH($\mathbf{B}', C', \emptyset, F \setminus \{m\},$ `count` $+ 1$)
  **else if** $C < C'$ and ($E = \emptyset$ or $F = \emptyset$) **then**
    **return B**
  **else**
    **return** INCH($\mathbf{B}, C, E \setminus \{m\}, F \setminus \{m\},$ `count`)

then contains derivatives of all branches and we select our modes by choice of $\mathbf{w}$. Computational issues may arise when computing the derivatives for inactive modes as they may not be defined, *e.g.* $1/\sqrt{y}$ when $y < 0$, but with the inclusion of appropriate safeguards in our generated code these can be safely mitigated. It should be noted that the artificial controls are not genuine degrees of freedom since we must have consistency between $(\mathbf{x}, \mathbf{u})$ and $\mathbf{w}$, *i.e.* $\mathbf{w}$ activates mode $m$ if and only if $(\mathbf{x}, \mathbf{u}) \in \mathcal{A}_m$.

If we consider the hybrid model as a computational sequence then the switching functions can be seen as conditions for the different branches of the intermediate variable evaluation. For example, the function

$$f(x,u) = \begin{cases} x^2 - u^2 & |x| \geq |u| \\ \frac{1}{u^2 - x^2} & |x| < |u| \end{cases}$$

can be written as the sequence

$$\begin{aligned} t_1 &\leftarrow u^2 \\ t_2 &\leftarrow x^2 \\ t_3 &\leftarrow t_2 - t_1 \\ &\texttt{if} \quad t_3 \geq 0 \ \texttt{then} \\ f &\leftarrow t_3 \\ &\texttt{else} \\ f &\leftarrow \frac{-1}{t_3} \end{aligned} \quad . \tag{7.1}$$

Using an artificial variable $w \in \{0, 1\}$ we can rewrite (7.1) as

$$\begin{aligned} t_1 &\leftarrow u^2 \\ t_2 &\leftarrow x^2 \\ t_3 &\leftarrow t_2 - t_1 \\ f &\leftarrow w t_3 + \frac{(w-1)}{t_3} \end{aligned} \tag{7.2}$$

which is equivalent to (7.1) when the following consistency condition is satisfied $w = 1 \Leftrightarrow |x| \geq |u|$.

For intermediate variables with multiple branches we need only introduce a single artificial variable $w \in \{\omega_1, \ldots, \omega_n\}$ whose range is equal in cardinality to the number of

branches $n$, *e.g.* the branching sequence

$$\begin{aligned} &\texttt{if} \rho_1(x,u) \leq 0 \ \texttt{then} \\ &\quad t_j \leftarrow \beta_1(x,u) \\ &\texttt{else if} \rho_2(x,u) \leq 0 \ \texttt{then} \\ &\quad t_j \leftarrow \beta_2(x,u) \\ &\quad \vdots \\ &\texttt{else if} \rho_n(x,u) \leq 0 \ \texttt{then} \\ &\quad t_j \leftarrow \beta_n(x,u) \end{aligned} \tag{7.3}$$

can be rewritten as

$$t_j \leftarrow \sum_{i=1}^{n} L_i(w)\beta_i(x,u) \tag{7.4}$$

where $L_i(w) = \prod_{m=1,m\neq i}^{n} \frac{w-\omega_m}{\omega_i-\omega_m}$ is the $i$th Lagrange basis polynomial so that $L_i(\omega_k) = \delta_{ik}$. Statements (7.3) and (7.4) are equivalent when $w = w_i \Leftrightarrow \rho_i(x,u) \leq 0$.

The above method allows us to compute exact derivatives despite the hybrid nature of the model and utilize the symSS formulation. Knowing $\mathbf{x}_0, \mathbf{U}$ we can compute the consistent sequence of artificial controls $\mathbf{W}$ by simply simulating the original hybrid model. Thus during the iterations within a solver we can maintain consistency as $\mathbf{U}$ is updated. For an MPC problem with fixed timestep, finding a consistent $\mathbf{W}$ is simpler than in the general setting of hybrid optimal control where one needs to search for the optimal switching points.

## 7.5 Case Studies

We have selected a variety of problems to both demonstrate and examine the QT strategy for hybrid NMPC problems. The first is the Lotka-Volterra fishing problem, a classical optimal control problem with a binary input, second is an electronic submarine problem that has one continuous input and one discrete input, third is an autonomous vehicle problem that has mixed-integer controls with an automatic transmission model, lastly is a diesel airpath problem that has a binary control with internal switches.

To evaluate the performance of the QT strategy we compare it to solutions generated using the Basic Open-source Nonlinear Mixed Integer programming (BONMIN) solver [101], chosen as a representative MINLP solver, and the convexification and relaxation (C&R) method of [102, 100]. For all solvers we pre-computed and optimized the appropriate

exact gradients and Hessians using the symSS formulation described in section 7.4. Thus for BONMIN and the C&R solvers this led to larger dimensional optimization problems since the discrete controls were relaxed and included as optimization variables. Constraints were handled using penalty functions for the QT strategy and the C&R method and were passed directly to the BONMIN solver. The BONMIN solver was run using default settings with either the Branch and Bound (BB) or Outer-Approximation (OA) algorithm. The details of these algorithms can be found in [101].

All simulations were run using MATLAB 2017b on a desktop with an Intel(R) Core(TM) i7-4790 CPU. BONMIN was run on MATLAB using the OPTI Toolbox v2.27 [103].

### 7.5.1 Lotka-Volterra Fishing Problem

The Lotka-Volterra Fishing Problem is a classic benchmark problem in controls that can be found in [104] and described on `mintoc.de`. The goal is to find an optimal fishing strategy to bring both predator and prey fish to a specific steady state. In the following formulation we regard the control to be binary, *i.e.* either no fishing or maximal fishing. The finite horizon NLP of the MPC is given by

$$\begin{aligned} \min_{\mathbf{U}} \mathcal{J}(\mathbf{U}; \mathbf{x}(0)) &= \sum_{k=1}^{H} \|\mathbf{x}(k) - \mathbf{x}^r\|^2 \\ \text{subject to} \quad \mathbf{x}(k+1) &= \mathbf{x}(k) + \Delta t l(\mathbf{x}(k), \mathbf{u}(k)), \\ k &= 0, \ldots, H-1 \\ \mathbf{U} &\in \{0,1\}^H \end{aligned}$$

where $H = 10, \Delta t = 0.01$s, and the target state is $\mathbf{x}^r = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$. The simulations are run for 12s and started at $\mathbf{x}[t_0] = \begin{pmatrix} 0.5 & 0.7 \end{pmatrix}^T$. The model arises from an explicit Euler discretization of the continuous time ODE model, $\dot{\mathbf{x}} = l(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} x_1(1 - x_2 - \frac{2}{5}u) \\ x_2(-1 + x_1 - \frac{1}{5}u) \end{pmatrix}$.

A dynamic programming (DP) solution was found using [105] where the defined state grid contained 1000 equidistant points on the interval $[0, 2]$. The C&R solution used the IPOPT solver (interfaced with MATLAB using the OPTI Toolbox v2.27) with `maxiter` = 10 and sum up rounding (SUR) strategy with a threshold of 0.5 [100]. Since there are no continuous optimization variables for this problem, no derivative information was needed for the QT methods only — the cost function itself.

In Fig. 7.1 we compare the trajectories of all solution methods. The parameter settings

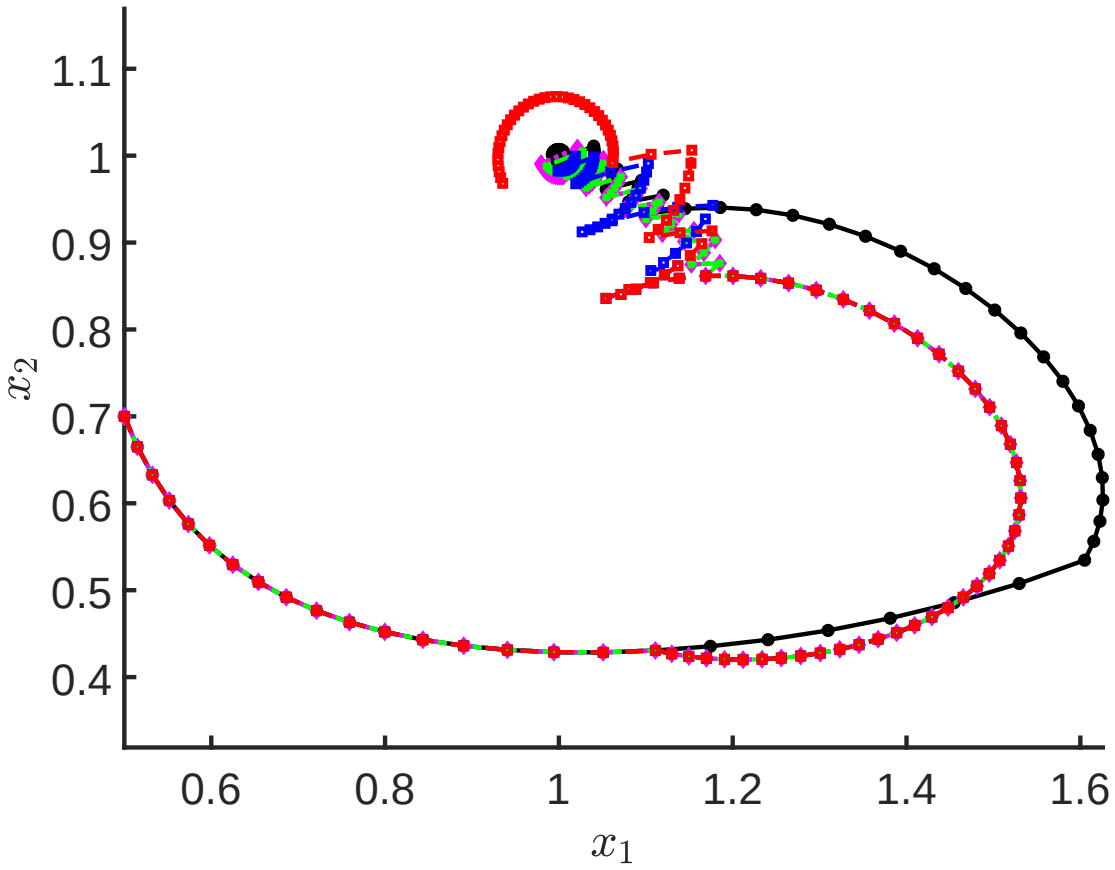



Figure 7.1: Sample solution trajectories for different solvers (black solid with circles: DP, red dashed with squares: IQT, blue dashed with squares: CWQT, dot-dashed green with triangles: C&R, magenta dotted with diamonds: BONMIN-OA, cyan dotted with diamonds: BONMIN-BB).

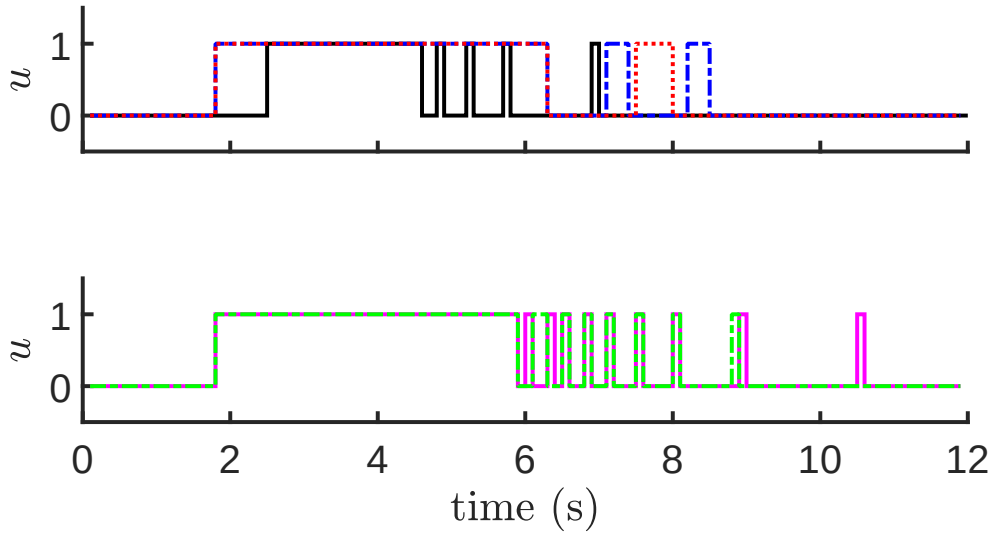



Figure 7.2: Sample control sequence for the different solvers (black solid: DP, red dotted: IQT, blue dashed: CWQT, dot-dashed green: C&R, magenta solid: BONMIN-OA, cyan dotted: BONMIN-BB). The plots are split into 2 to ease readability.

of the QT solutions are IQT: $s = 3, l = 1, p = 3$ and CWQT: $s = 3, l = 1, r = 5$. The two BONMIN solutions are equal and overlap entirely. We can see all solutions approach the point $\mathbf{x}^r$.

In Fig. 7.2 we display the control sequences of the different solutions. We can see the DP solution, with its global knowledge, is able to switch the least and achieve the best performance. All NMPC solutions first switch at the same timestep, before 2s, whereas the DP solution switches later. Surprisingly, the BONMIN and C&R solutions arrive at very similar switching behaviour as can be seen in the second plot.

In terms of performance, the DP solution performed best (with an overall cost of 1.55 where the overall cost is defined by $\Delta t \sum \|\mathbf{x}(k) - \mathbf{x}^r\|^2$ where the sequence $\mathbf{x}(k)$ is the solution over the entire simulation). Next best was the C&R and BONMIN solutions which were 34% worse (at 2.08) followed by the CWQT, IQT solutions which were 35% (2.10) and 36% (2.11) worse, respectively. Relative to the DP optimum the QT solutions were at most 2% worst than the BONMIN solution.

The BONMIN and C&R solutions switch 20 times over the 12s simulation and the DP only 10. The CWQT and IQT solutions switch 10 and 6 times, respectively. Thus the QT solutions reduce control switching by at least 2 (compared to BONMIN) with a small

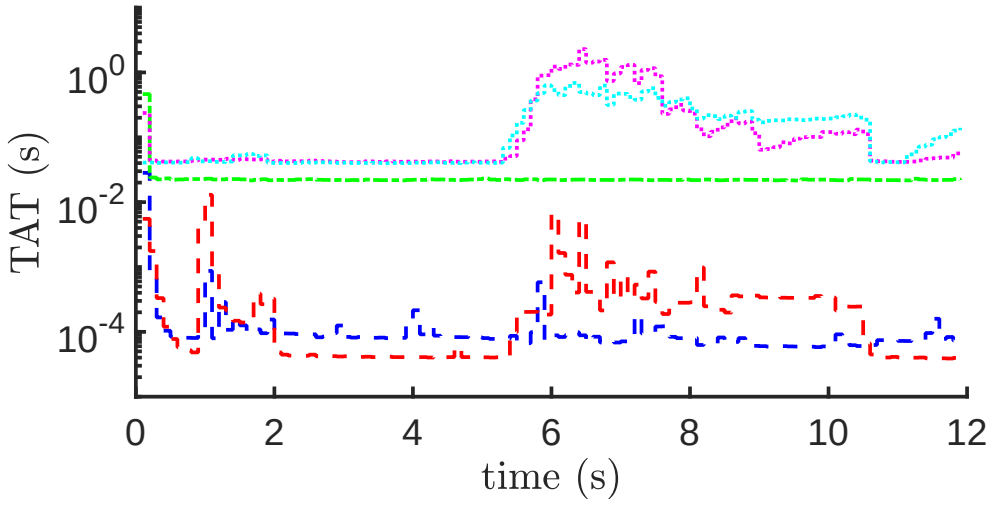



Figure 7.3: Sample TATs of the solvers (black solid: DP, red dashed: IQT, blue dashed: CWQT, dot-dashed green: C&R, magenta dotted: BONMIN-OA, cyan dotted: BONMIN-BB).

performance degradation.

In Fig. 7.3 we display a sample of the turnaround times (TATs) of the solvers over a simulation. The QT solutions have at least an order of magnitude smaller average TATs. The BONMIN solutions are the slowest. This large gap in computing time can justify the use of the QT strategy despite the small performance degradation.

In the following we examine the role of the parameters on the performance of the QT solutions, as well as the greedy QT search. Given $H = 10$, we consider $s, p, r \in \{1, \ldots, H-1\}, l \in \{1, \ldots, H-2\}$. Over this parameter range the best performance of CWQT using the greedy search is 2.0977, whereas the best performance of CWQT using full CWQT enumeration is 2.0755. Similarly, for IQT the best performance using the greedy search is 2.0979 and using full IQT enumeration is 2.0973. Thus in the case of CWQT the greedy algorithm performs only 1.07% worse and in the case of IQT it performs only 0.03% worse. Thus the use of the greedy optimization scheme does not significantly degrade the performance over a naive full enumeration approach. In the following, we utilize the greedy QT scheme by default.

In Fig. 7.4 we display the cost performance of CWQT relative to the best performance over the set of parameters. As can be seen, performance degrades steadily with larger $l$, is largely unaffected by $r$ and is worst for $s = 1$.

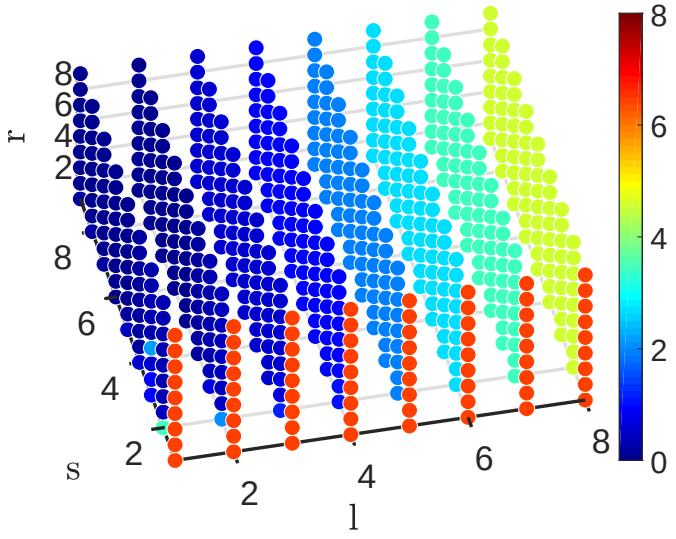



Figure 7.4: Relative CWQT performance (%).

In Fig. 7.5 we display the cost performance of IQT relative to the best performance over the set of parameters. Performance is best for larger $s$ and $p$ with small $l$. Interestingly large $l$ does not degrade performance steadily like the CWQT case. Again performance is worst when $s = 1$ and is also poor for $p = 1$.

In Figs. 7.6 and 7.7 we display the maximum number of NLP subproblems solved (an architecture-free measure of TAT) during a timestep over the set of parameters for CWQT and IQT, respectively. Here we can see $l$ has a large effect on this value. Only for small values of $l$ do $s, r, p$ have a significant impact.

In Figs. 7.8 and 7.9 we display the number of switches that occurred over the simulation for CWQT and IQT, respectively We see that for larger $l$ chatter is kept to a near minimum, independent of $s, r, p$. When $l = 1$ chatter increases most dramatically especially for large $s$. In CWQT this relation is most pronounced.

To minimize TAT, chatter and cost we want to pick a parameter set that approximately minimizes all 3 plots. From the trends present in these plots it appears that small values of all parameters with $s > 1$ are near optimal depending on the relative importance of TAT, chatter and cost.

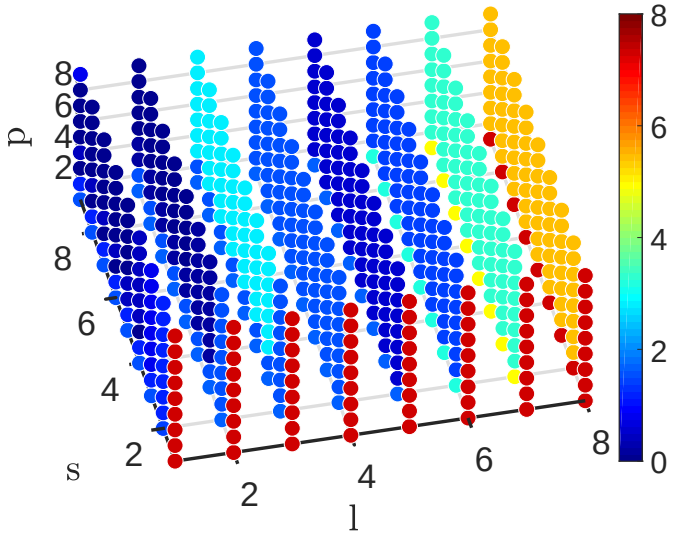



Figure 7.5: Relative IQT performance (%).

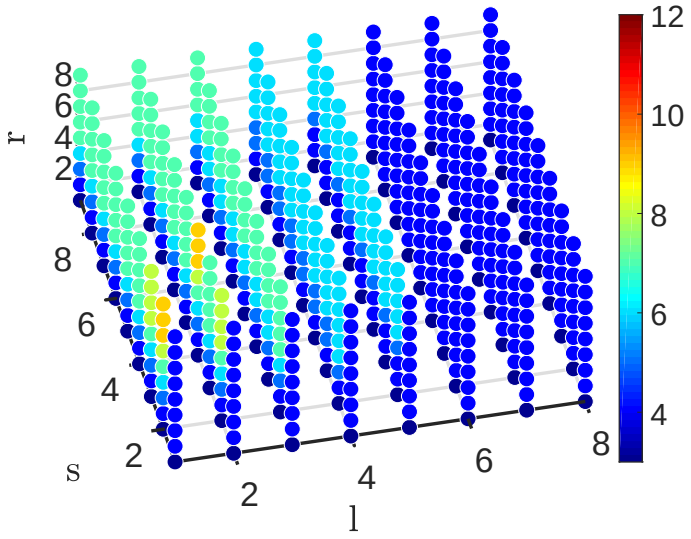



Figure 7.6: CWQT max number NLP subproblems.

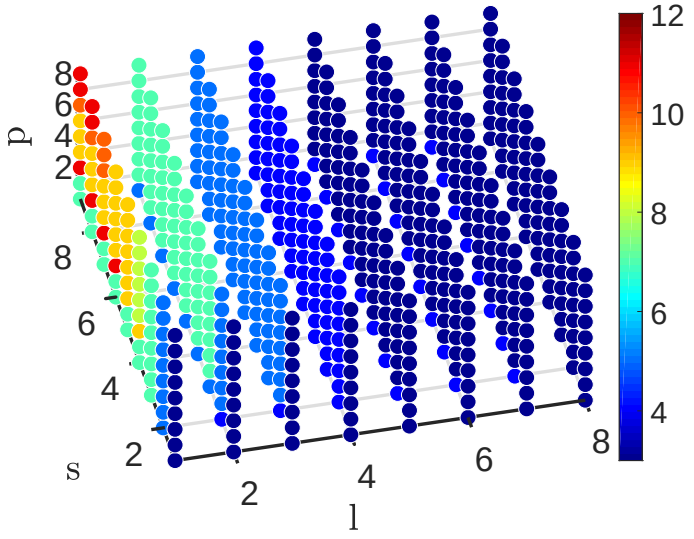



Figure 7.7: IQT max number NLP subproblems.

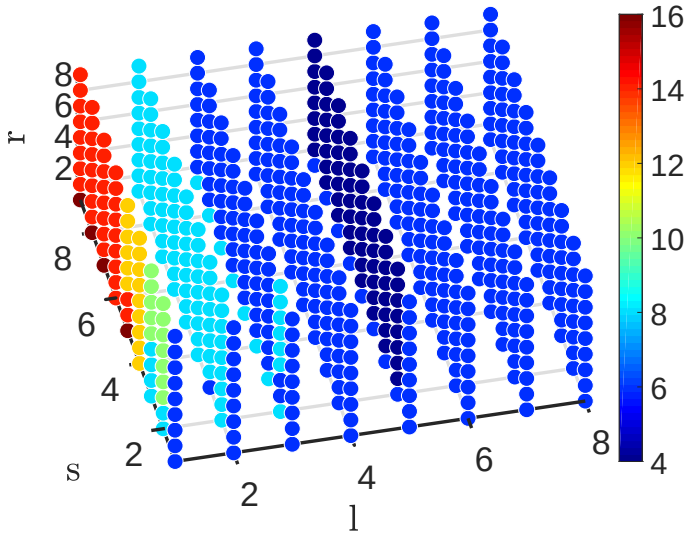



Figure 7.8: CWQT number of realized switches.

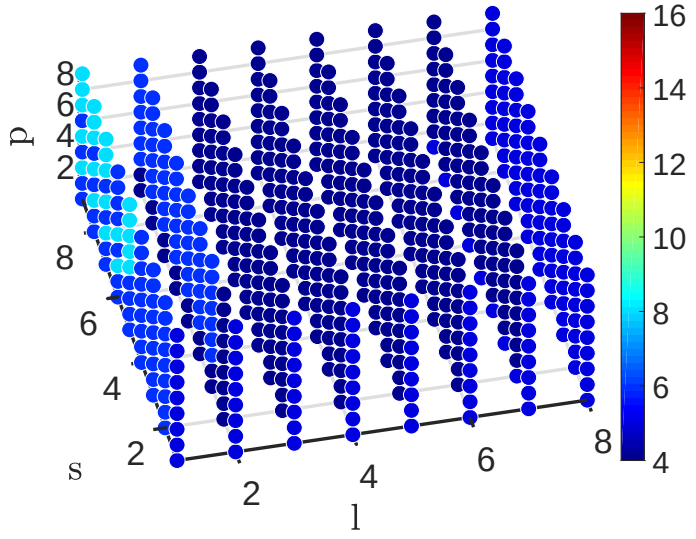



Figure 7.9: IQT number of realized switches

### 7.5.2 Diesel-Electric Submarine Problem

The goal of the controller in this problem is to find an optimal strategy to run the generators of the submarine so that the state of charge (SOC) of the battery is maintained at a reasonable level while minimizing the noise generated and ensuring the submarine travels a certain distance. The submarine is battery powered with three diesel generators for recharging. Each generator operates in 2 modes: normal and supercharged mode (which regenerates faster but more loudly).

Details of the dynamic model can be found in [106, 107], which we describe here briefly. The model is described by the differential equation

$$\dot{\mathbf{x}} = S(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} R(\mathbf{x}, \mathbf{u}) - D(\mathbf{u}) \\ u_1 \end{pmatrix} \tag{7.5}$$

where $x_1$ is the battery SOC and $x_2$ is the distance travelled by the submarine, $u_2 \in \{0, 1, \ldots, 6\}$ is the operating mode of the submarine and $u_1$ is the speed of the submarine. In (7.5) the recharge rate and discharge rate are given by

$$R(\mathbf{x}, \mathbf{u}) = \frac{2 \times 10^4}{x_1 + 100} a(u_2) \quad \text{and}$$
$$D(\mathbf{u}) = \begin{cases} \frac{u_1^3}{120} + 8 & u_2 \neq 0 \\ \frac{u_1^3}{80} + 5 & u_2 = 0 \end{cases},$$

respectively, where $a(u_2)$ is the $u_2^{th}$ element of $a = [0, 0.3564, 0.333, 0.5128, 0.6666, 0.7692, 1]$ (beginning at element 0) is the fraction of full power produced by the generators given the operating mode $v$. Furthermore, based on physical constraints and desired submarine operation the following constraints are enforced: $u_1 \in [5, 60]$ (km/h) and $x_1 \in [20, 100]$.

To formulate our finite horizon optimization problem we discretize (7.5) using the explicit Euler method with constant timestep $\Delta t$. At timestep $t_i$ the optimization problem

is given by

$$\begin{aligned}
\min_{\mathbf{U}} \mathcal{J}(\mathbf{U};\mathbf{x}(0)) &= \sum_{k=1}^{H}[\alpha(x_1(k)-100)^2+\sigma\rho_p(x_1(k))^2] \\
&\quad + \sum_{k=0}^{H-1}[\beta\ln(n(u_2(k))+1)+\upsilon\rho_p(u_1(k))^2] \\
&\quad + \gamma(x_2(H+1)-L[i])^2 \\
\text{subject to} \quad \mathbf{x}(k+1) &= \mathbf{x}(k)+\Delta t S(\mathbf{x}(k),\mathbf{u}(k)), k=0,\ldots,H-1 \\
\mathbf{X} &\in ([20,100]\times\mathbb{R})^H \\
\mathbf{U} &\in ([5,60]\times\{0,1,\ldots,6\})^H
\end{aligned}$$

where $H=20$, $\alpha=10^3, \beta=2\times10^4, \gamma=2\times10^4, \sigma=1, \upsilon=100, L[i]=840\frac{i+H}{t_f/\Delta t-H}, \Delta t=0.01\text{hr}, t_f=24$ hr and $n(u_2)$ is the $u_2^{th}$ element of $[0,10,15,20,30,30,45]$ (beginning index at 0) is the noise produced by the generators of an operating mode during recharge. To enforce the box constraints we used the penalty function (3.3) presented in Chapter 3. The simulation is begun at full battery charge $\mathbf{x}[t_0]=\begin{pmatrix}100 & 0\end{pmatrix}^T$.

We apply the QT strategy by introducing an artificial variable $w\in\{0,1\}$ such that $w=0\Leftrightarrow v=0$. Then we replace $D(\mathbf{u})$ in the model by the sum $D'(\mathbf{u},w)=w\left(\frac{u_1^3}{120}+8\right)+(1-w)\left(\frac{u_1^3}{80}+5\right)$. To apply BONMIN we let relaxed $u_2$ and 'smoothed' $D$ to be $D''=\tanh(5u_2)(\frac{u_1^2}{120}+8)+(1-\tanh(5u_2))(\frac{u_1^2}{80}+5)$. We also used polynomial interpolations of functions $a,n$. To apply the C&R method we used a SUR strategy as well as polynomial interpolations of functions $a,n$.

In Fig. 7.10 we compare the trajectories of the solvers. In these simulations we have selected the following parameters, CWQT: $s=3,l=1,r=2$ and IQT: $s=2,l=1,p=2$. The BONMIN and C&R solutions try and maintain near maximum SOC while the QT solutions are more flexible. In terms of final distance travelled the C&R solution performs best being only 7.00km off target (a 0.83% relative error) while the CWQT performs worst being 29.69km off the target distance of 840km (a 3.53% relative error) or 2.7% worse.

In Fig. 7.11 we can see the highly variable control strategies of the different solvers. The QT solutions exhibit regular periodic switching behaviour that is very similar qualitatively to the solutions in [107] which use the MISER3.2 software based on [108]. This solution consists of periods of maximal speed with rapid charging followed by periods of low speed and no charging. The BONMIN and C&R solutions exhibit rapid switching of one control

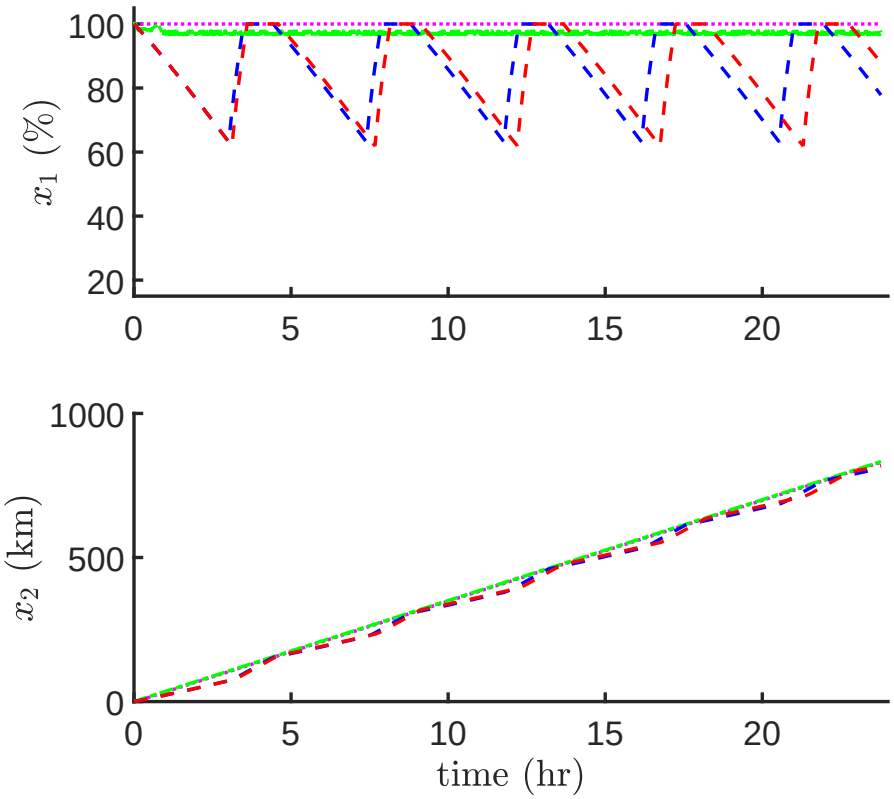



Figure 7.10: Solver trajectories (dotted pink: BONMIN-BB, solid green: C&R, dashed blue: CWQT, dashed red: IQT).

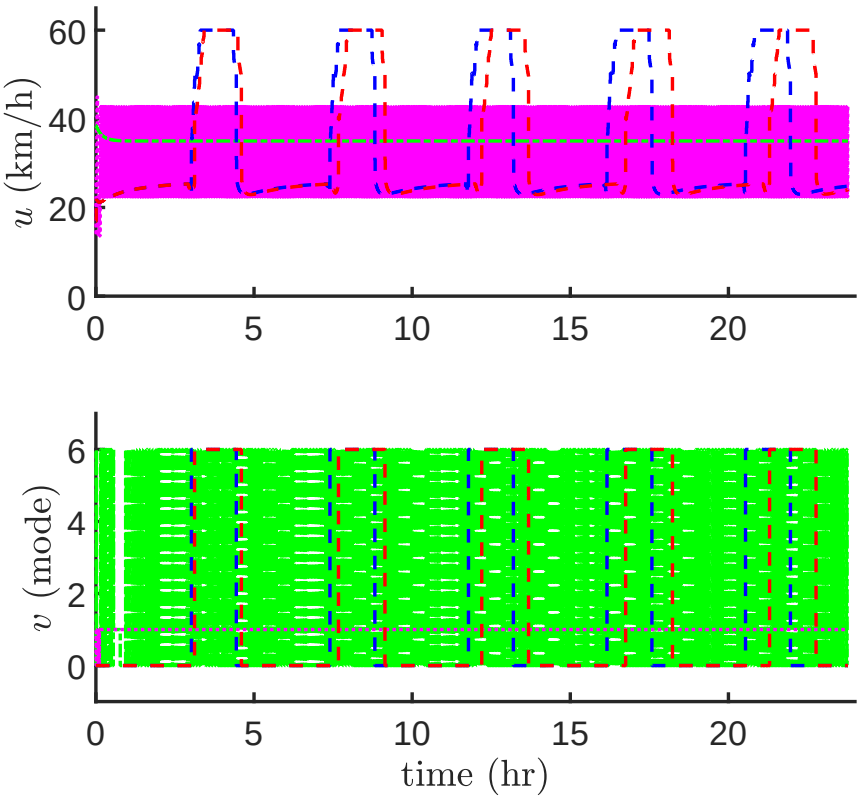



Figure 7.11: Solver performance (dotted pink: BONMIN-BB, solid green: C&R, dashed blue: CWQT, dashed red: IQT).

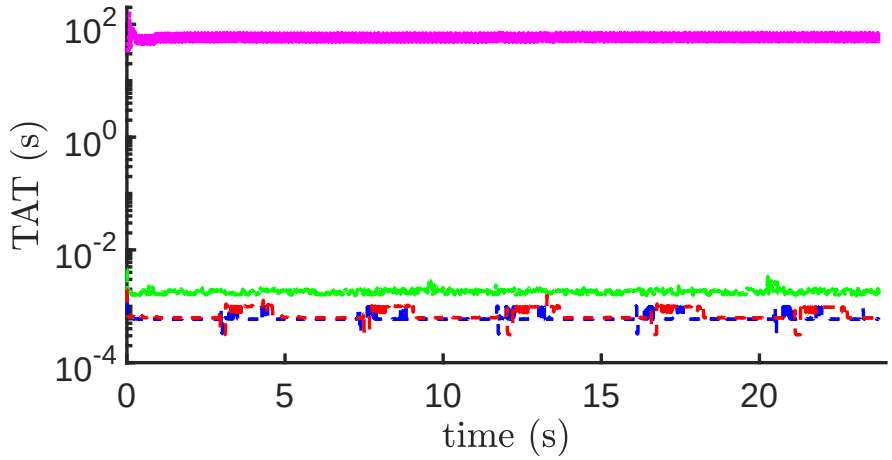



Figure 7.12: Solver TATs. (dotted pink: BONMIN-BB, solid green: C&R, dashed blue: CWQT, dashed red: IQT)

while holding the other near constant. Interestingly, they do this to opposite controls. However, both are highly undesirable since rapid switching or chatter contributes to a plant's degradation.

In Fig. 7.12 we display the TATs of a sample simulation. In this figure the mean TATs are: 57.16s for BONMIN, 1.931ms for C&R, 0.6237ms for CWQT and 0.7670ms for IQT. The TATs of the QT solutions are again smaller than the other methods.

In the following we briefly discuss some of the effects the parameters have on the QT solutions. Firstly, it was found the parameter $l$ had no discernible impact on performance for both QT strategies. This is likely due to the low switching frequency of the QT solutions. In the following we display the results for the CWQT strategy over the range $s, r \in \{1, \ldots, H\}$ for $l = 1$.

In Fig. 7.13 we display the relative increase in error over the simulation compared to the smallest value (error here is given by $\Delta t \sum \alpha(x_1(k) - 100)^2 + \beta \ln(n(u_2(k)) + 1)$ where the sum is taken over all steps of the simulation). In Fig. 7.14 we display the relative terminal position error, $|x_2(t_f) - L|/L$ where $L = 840$km. From these we can see $r$ has minimal effect on performance. Only $s$ has an impact, particularly at smaller values where we see a tradeoff on terminal position accuracy for cost. In Fig. 7.15 we display the maximum number of NLP subproblems per timestep over the simulation. Again $r$ does not play a significant role. In terms of $s$ we see an almost linear trend in number.

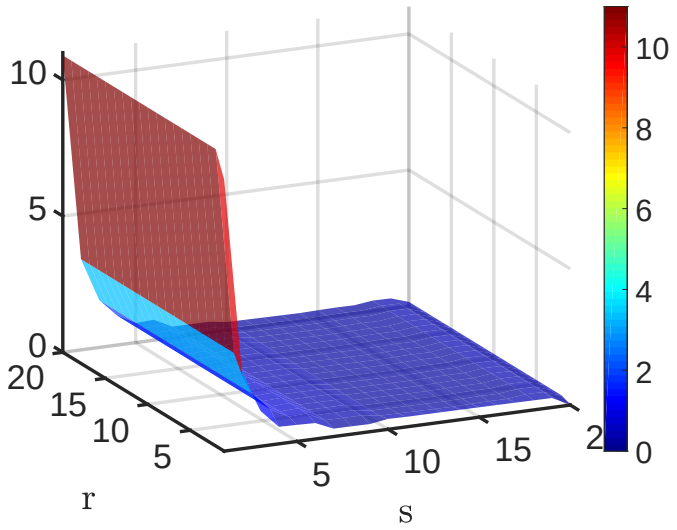



Figure 7.13: CWQT Relative Error Increase (%).

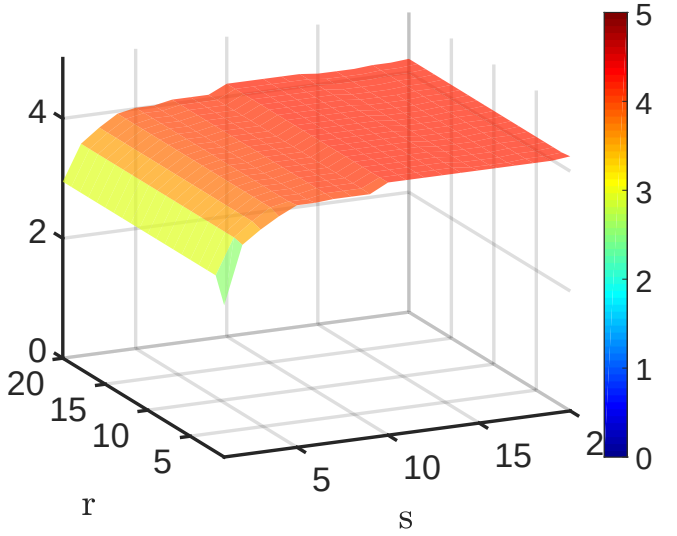



Figure 7.14: CWQT Relative Terminal Error (%).

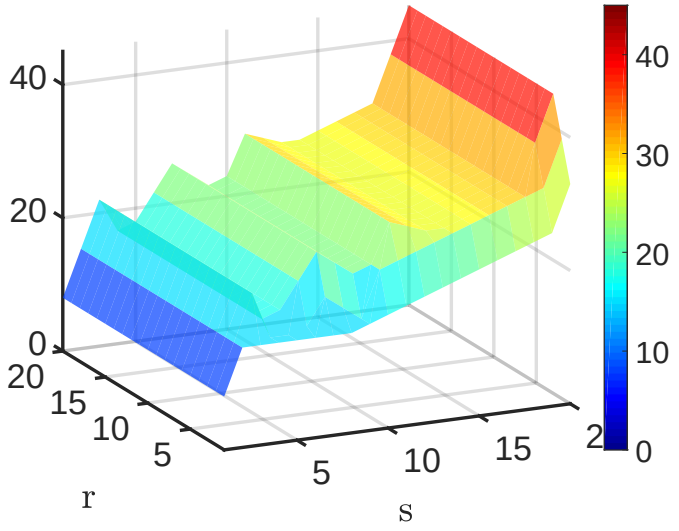



Figure 7.15: CWQT maximum number NLP subproblems per timestep.

For the IQT strategy all parameters $s, l, p$ were found to have insignificant impacts on performance. The relative errors and maximum number of NLP subproblems were all effectively constant. In terms of cost the CWQT strategy performed about 9% better. For both strategies there was no parameter effect on the number of switches present in the solution.

For this problem it is interesting to see that the QT strategy is able to find a desirable strategy that the competing methods could not. The low frequency switching of the found solution largely negated the effects of the parameters (except $s$ for CWQT) but indicates that for problems with a natural low frequency of switching, the QT solution can find it quickly. For this problem it is interesting to see that the QT strategy is able to find a desirable strategy that the competing methods could not. The low frequency switching of the found solution largely negated the effects of the parameters (except $s$ for CWQT) but indicates that for problems with a natural low frequency of switching, the QT solution can find it quickly.

### 7.5.3 Diesel Airpath Model

The main goal of this controller is to track a reference output trajectory where $\mathbf{y} = \begin{pmatrix} p_{int} & \phi_{EGR} \end{pmatrix}^T$ are the intake manifold pressure and exhaust gas recirculation (EGR) rate, are the outputs of interest. Tracking these two outputs allows the engine to meet the driver's demands while satisfying emission requirements. The diesel airpath (DAP) model under consideration is a 9 state, nonlinear, physics based-model with 3 continuous controls, 1 binary control, 2 disturbance inputs and 2 internal switches. The disturbance inputs of the model were held constant in our simulations. More information on the model and its derivation can be found in [53] and the references therein.

A further objective of the controller is to minimize the rate of change of the controls $\mathbf{u} = \begin{pmatrix} \theta & u_{EGR} & u_{VGT} & v \end{pmatrix}^T$, throttle command [% closed], EGR valve command [% open], variable-geometry turbocharger (VGT) command [% normalized], and EGR cooler bypass setting while satisfying the control bounds $\theta \in [0, 100], u_{EGR} \in [0, 70]$, $u_{VGT} \in [40, 80]$ and $v \in \{0, 1\}$. The problem cost is given by

$$\mathcal{J}(\mathbf{U}) = \sum_{k=0}^{H-1} \|\mathbf{y}(k) - \mathbf{y}^r(k)\|_{\mathbf{Q}}^2 + \|\Delta\mathbf{u}(k)\|_{\mathbf{R}}^2 + \|\rho_p(\mathbf{u}(k))\|_{\mathbf{W}}^2$$

where, $y^r$ is a reference output trajectory, $\Delta\mathbf{u}(k) = \mathbf{u}(k) - \mathbf{u}(k-1)$ $(\Delta\mathbf{u}(0) = 0)$, $\mathbf{Q} = \text{diag}\begin{pmatrix} 1 & 10^4 \end{pmatrix}$, $\mathbf{R} = \text{diag}\begin{pmatrix} 10 & 10 & 10 & 0 \end{pmatrix}$, $\mathbf{W} = \text{diag}\begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}$. The control bounds are enforced using the penalty function (3.3) from Chapter 3 evaluated entry-wise for vector-valued arguments. This problem has the same form as the control problem in Section 5.3.2. In this study however, we consider a different plant model that is hybrid with constant external inputs.

In Fig. 7.16 we display the tracking performance, as well as the internal switching dynamics of the model for a sample simulation. We note that only a single internal switch was activated over this simulation. Here we used a CWQT strategy as a representative for the QT strategy since both strategies performed very similarly. It is clear from the figure that the QT strategy performed best in tracking error compared to the C&R solution. The values of this difference are captured in Table 7.1 which summarizes the results of 6 different simulations (including the one shown in Fig. 7.16). From Table 7.1 we see that the QT method is more than an order of magnitude better at tracking than the C&R strategy.

In Fig. 7.17 we display the controls of the simulation displayed in Fig. 7.16. From this we can see that the C&R strategy is unable to take advantage of the binary input (this was tested for a variety of SUR thresholds). We note that the solutions are similar when the

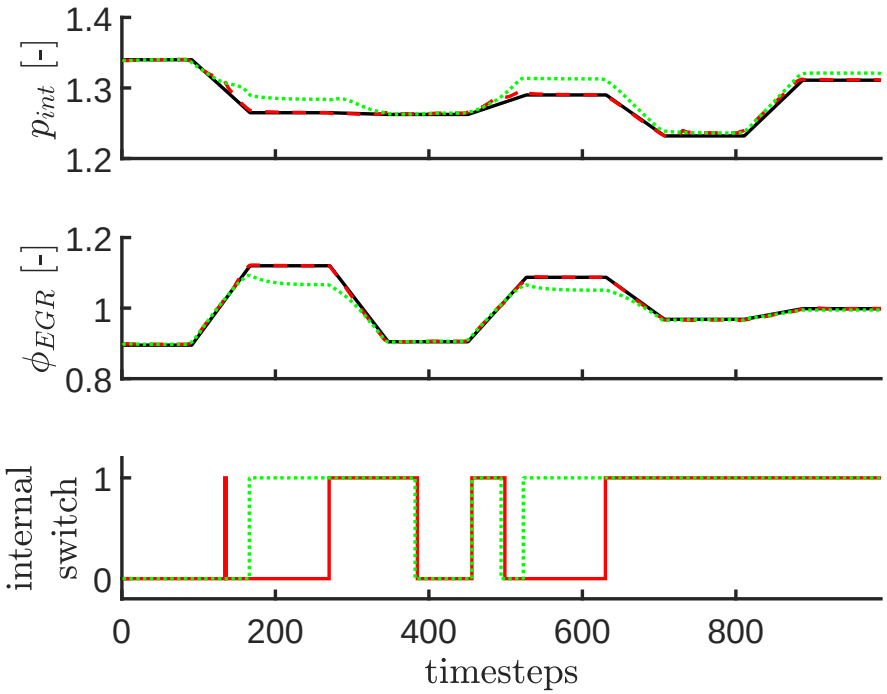



Figure 7.16: Solver performance (solid black: reference, dotted green: C&R, dashed/solid red: QT). All output values have been normalized and offset.

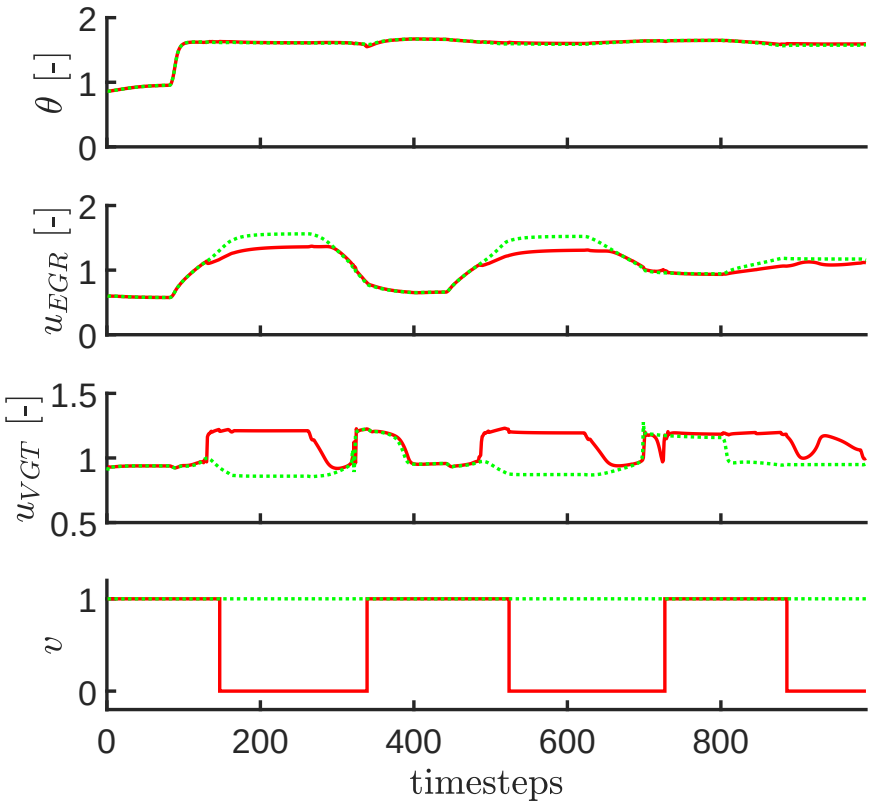



Figure 7.17: Controls. (dotted green: C&R, solid red: QT). All values have been normalized and offset.

| | $p_{int}$ [kPa] | $\phi_{EGR}$ [-] |
|---|---|---|
| QT | 0.2125 | 0.0012 |
| C&R | 10.9865 | 0.0719 |

Table 7.1: Mean RMS Tracking Error of 6 simulations

bypass mode is equal. It is a shortcoming of the C&R strategy that a rounding strategy is unable to take advantage of the binary input.

Similar to the electric submarine problem the parameters did not effect the QT solutions significantly. This again is largely due to the relatively slow switching frequency of the problem. We do not report TATs for this problem since we could not get a fair comparison of the two approaches. MATLAB was unable to convert the Hessian of the C&R strategy to a MEX file.

### 7.5.4 Autonomous Vehicle Problem

We apply the QT approach to an autonomous vehicle control problem that has two discrete inputs: a binary pedal input, to ensure the exclusive action of the throttle or brake, and a gear selection input. Previous work on mixed-integer optimal control of vehicles has only considered the gear selection as an integer control, *e.g.* [109] developed a cruise-controller with discrete gear input using dynamic programming combined with preprocessing, [110] developed a predictive gear selection and cruise controller for a heavy truck using a BB method and [102], [111] and [75] consider the optimal control of a vehicle with discrete gear input using a convexification and relaxation method, a variable time transformation approach, and a BB method, respectively.

In this study we demonstrate the use of QTs in the control of an autonomus vehicle. We consider two hybrid models based on the one given in Appendix A. In the first case we suppose the model has a manual transmission and so the gear input is an integer control. In the second case we provide a rudimentary automatic transmission model based on the vehicle speed so that the gear input becomes an implicit switch. For both cases we introduce a binary control $b \in \{0, 1\}$ to enforce exclusive pedal input and a normalized pedal value $\theta \in [0, 1]$. From these we can compute the brake force $F_B$ and throttle $\phi$ inputs of the reference model via the following mappings $F_B = 15000(1 - b)\theta$ and $\phi = b\theta$. Thus the control inputs for this problem is given by $\mathbf{u} = \begin{pmatrix} w_\delta & \theta & b & \mu \end{pmatrix}$.

The driving scenario is set up to mimic moderately aggressive urban driving and is 238.5s ($\approx$ 4 min) in duration. The reference path and speed is plotted in Fig. 7.18.

The goal of the controller is to follow a reference trajectory while minimizing the inputs and their rates of action. The cost function is given by

$$\mathcal{J}(\mathbf{U}; \mathbf{x}(0)) = \sum_{k=1}^{H} \|\mathbf{x}(k) - \mathbf{x}(k)^r\|_{\mathbf{Q}}^2 + \|\mathbf{v}(k-1)\|_{\mathbf{R}}^2 + \|\Delta\mathbf{v}(k)\|_{\mathbf{R}'}^2 + \sum_{k=1}^{H} \|\rho_p(\mathbf{u}(k-1))\|_{\mathbf{W}}^2 \quad (7.6)$$

where $H = 10$ is the number of timesteps in the horizon, $\mathbf{x}^r$ is the reference trajectory, $\mathbf{v} = \begin{pmatrix} w_\delta & F_B & \phi \end{pmatrix}^T$ are the continuous inputs we wish to minimize and $\Delta\mathbf{v}(k) = \mathbf{v}(k) - \mathbf{v}(k-1)$ (with $\Delta\mathbf{v}(H) = 0$) are the input rates. The weighting matrices are $\mathbf{Q} = \text{diag}\begin{pmatrix} 10^4 & 10^4 & 2 \times 10^4 & 0 & 0 & 100 & 0 \end{pmatrix}$, $\mathbf{R} = \text{diag}\begin{pmatrix} 500 & 0 & 200 \end{pmatrix}$ and $\mathbf{R}' = \text{diag}\begin{pmatrix} 200 & 10^4 & 10^4 \end{pmatrix}$. The states over the horizon are defined recursively using the explicit Euler method $\mathbf{x}_{k+1} = \mathbf{x}(k) + \Delta t \Phi(\mathbf{x}(k), \mathbf{v}(k))$ where $\Delta t = 0.05$s is the fixed timestep and $\Phi$ is the model's vector field (see: Appendix A). We set $p = 4$ and $\mathbf{W} = \text{diag}\begin{pmatrix} 10 & 100 & 0 & 0 \end{pmatrix}$ in our simulations.

We now turn our attention to the first case, where there are two integer controls — the pedal choice $b$ and gear input $\mu$. We compare our results to the solution found by the Basic

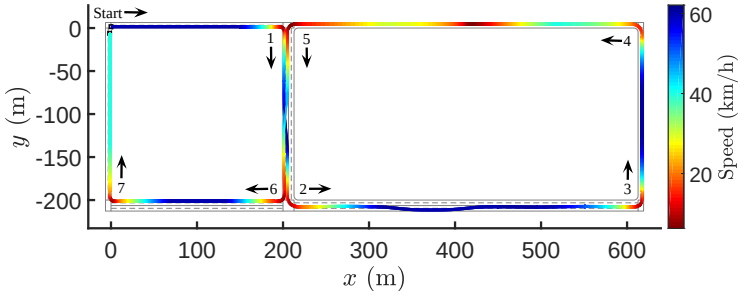



Figure 7.18: The reference path follows a figure eight that begins and ends near $(0, 0)$. The corners are labelled in the order they are encountered. Between corners 1 & 2 there is a lane change to the left, between corners 2 & 3 there is a double lane change, between corners 4 & 5 there is a slow down maneuver and between corners 5 & 6 there is a lane change to the right.

Open-source Nonlinear Mixed Integer programming (BONMIN) solver [101]. This solver is chosen as a representative of existing MINLP methods. At each timestep BONMIN was applied to $\mathcal{J}$ with bounds on $\mathbf{u}$ and linear inequalities enforcing sequential gear changes as constraints. We also used a fourth-degree polynomial interpolation of the transmission relation $i_g(\mu), \mu \in [1, 5]$ for the relaxed subproblems. BONMIN was implemented with all default parameters as a NLP based BB algorithm, which we denote BONMIN-BB. Other BONMIN algorithms, such as Outer-Approximation, were tried with default parameters but they all failed at some stage of the simulation with an exitflag indicative of a solver error or unbounded subproblem.

Using Maple, the objective function, its gradient and Hessian were generated and optimized as standalone functions. These functions were then converted to MEX functions that we then used in our fast NLP method and we passed to BONMIN. All simulations were run using MATLAB 2017b on a desktop with an Intel(R) Core(TM) i7-4790 CPU. BONMIN was run on MATLAB using the OPTI Toolbox v2.27 [103].

We select one parameterization of each of the QT strategies to compare with the BONMIN-BB solution. The first is a CWQT with parameters $s_b = 1, r_b = 4, s_\mu = 3, l_\mu = 1, r_\mu = 3$ and the second is an IQT with parameters $s_b = p_b = 1, s_\mu = p_\mu = 3, l_\mu = 3$. The subscripts denote which integer control the parameters are determining. In Figures 7.19 and 7.20 we show an overview of the simulations over the entire scenario. Firstly, we note the BONMIN-BB solution is remarkably similar in the switching times of the throttle/brake input to the QT strategies. However, the gear selection differs more dramatically between

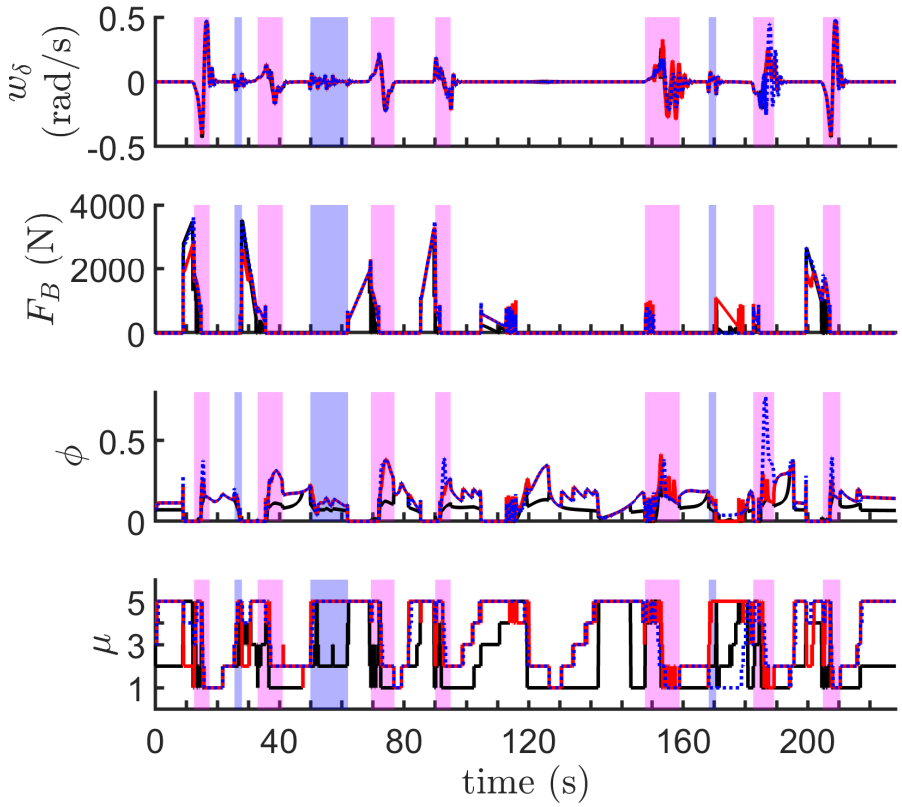



Figure 7.19: Control inputs for the urban driving scenario. The solid black, solid red and dotted blue lines are the BONMIN-BB, CWQT and IQT solutions.The pink and blue patches highlight periods of turning and lane changing, respectively.

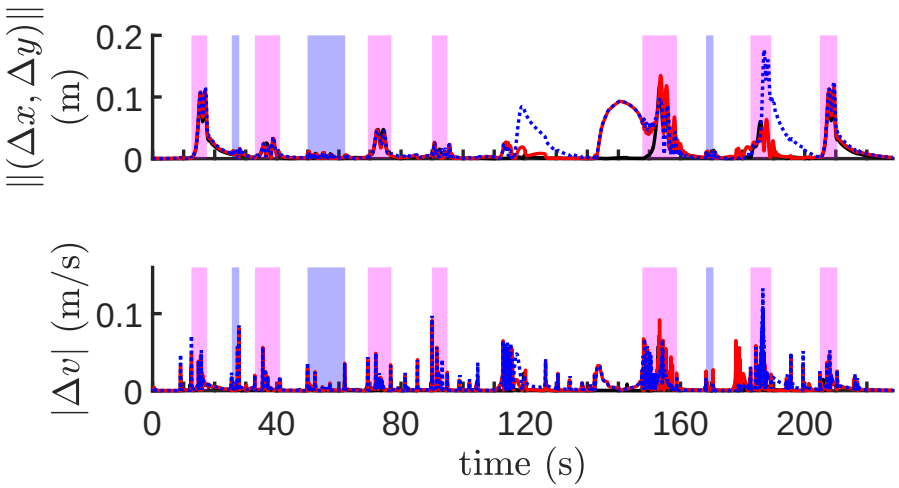



Figure 7.20: Position and velocity tracking errors.

solutions. This is to be expected since the gear selection $\mu$ is not controlled explicitly by our choice of objective function but is implicitly influenced by the desire to minimize $\phi$. Greatest chatter of the BONMIN-BB solution occurs about the throttle/brake switches. We can see that in all solutions the throttle to brake switches mostly occur going into a corner and the brake to throttle switches mostly occur halfway through a corner, much like a human driver. The position error of all solutions spike during cornering.

In Figure 7.21 we zoom in on the control inputs over the 10-50s period. From this figure we can more clearly see a number of results. Firstly, the BONMIN-BB solution switches more often than the QT solutions; however it is more efficient as it uses less total throttle input. This is partly due to the controller tuning which placed much greater cost on tracking error than control effort. Around the 120-150s period we see that the BONMIN-BB solution is able to rapidly shift down and up during the straight slow-down maneuver and thus be more efficient in its use of throttle. However, the BONMIN-BB solution at some points switches gears 19 times (the maximum possible) within a one second span; meanwhile the CWQT and IQT solutions switch gears at most 7 and 3 times within a one second span, respectively. The QT strategy reduces the chattering behaviour at the expense of efficiency.

Further, looking at Figure 7.20 we can see that the BONMIN-BB solution has better tracking accuracy. The BONMIN-BB solution is never more than 11.94 cm from its target position. In comparison, the IQT trajectory (which performs worst) deviates as much

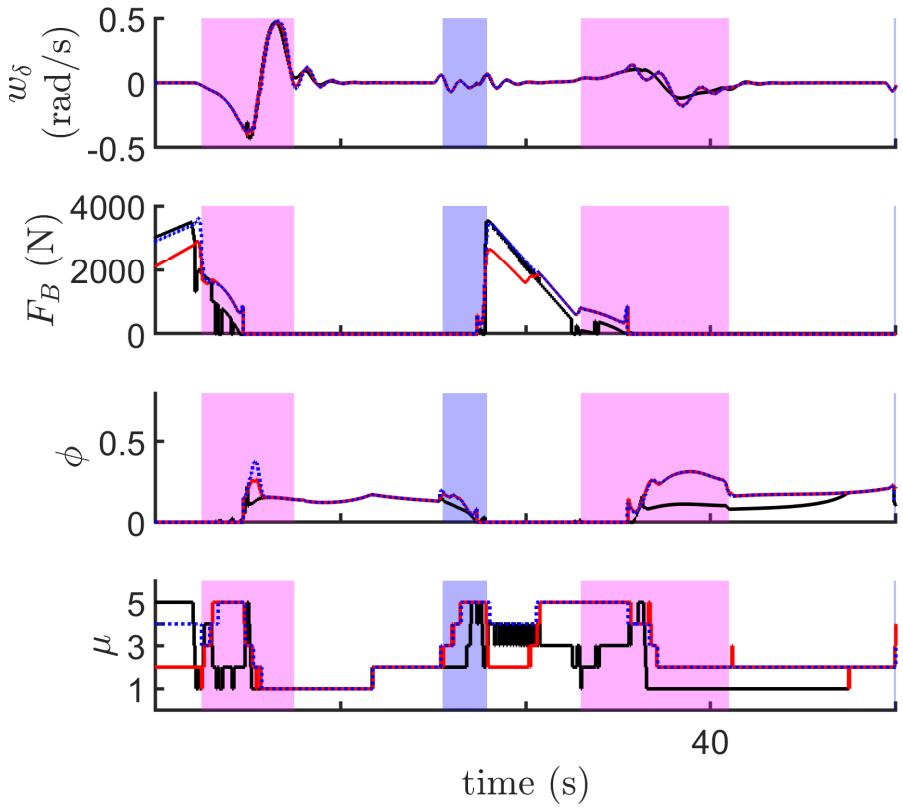



Figure 7.21: Control inputs close-up over the 10-50s timespan.

Table 7.2: NMPC Accuracy

| | mean $\|(\Delta x, \Delta y)\|$ (cm) | mean $\|\Delta v\|$ (cm/s) |
|---|---|---|
| BONMIN-BB | 0.90 | 0.16 |
| CWQT | 1.61 | 0.49 |
| IQT | 2.15 | 0.57 |

Table 7.3: NMPC Computation Times

| | Total (s) | Max Turnaround (s) |
|---|---|---|
| BONMIN-BB | 6098.1 [-] | 115.6 [-] |
| CWQT | 18.8 [$> 300\times$] | 0.0084 [$> 13000\times$] |
| IQT | 12.1 [$> 500\times$] | 0.0057 [$> 20000\times$] |

as 17.64 cm from target. However, relative to the lane width (3.2 m) this is only a 2% increase in position error. Similarly, BONMIN-BB is at most 2.5% better in terms of relative velocity error. These results are expected since BONMIN-BB relies on the IPOPT solver for its NLP subproblems, which we expect to find better solutions than our fast NLP solver that uses a fixed number of Newton iterations. We summarize these results in Table 7.2.

In Table 7.3 we summarize the computation time results along with the relative increase in computing speed (shown in square brackets) for the entire scenario and maximum turnaround time. This is where there is greatest difference in controller behaviour. The QT approaches run orders of magnitude faster than BONMIN-BB and run faster than real-time on our computing architecture that uses an Intel(R) Core(TM) i7-4790 CPU. Most crucially we can see the importance of being able to bound the MINLP solution cost since existing solution methods may require an exponential effort, dramatically increasing turnaround times. Overall, given the relatively small decrease in tracking performance, these timing results are excellent.

We know turn our attention to the second case where there gear is selected via a state dependent switch, thus removing an external switch and introducing an internal one. Inspired by [112] we place limits on the engine speed for particular gears. Because of the rudimentary nature of the motor model this is equivalent to setting velocity limits on the gears because the model contains a proportional relation between engine speed and velocity. In our model, gear $\mu$ is active over the velocity interval $[20(\mu-1), 20\mu]$ (in km/h), *e.g.* gear 3 is active over the $40-60$ km/h range. This is akin to setting an upper limit

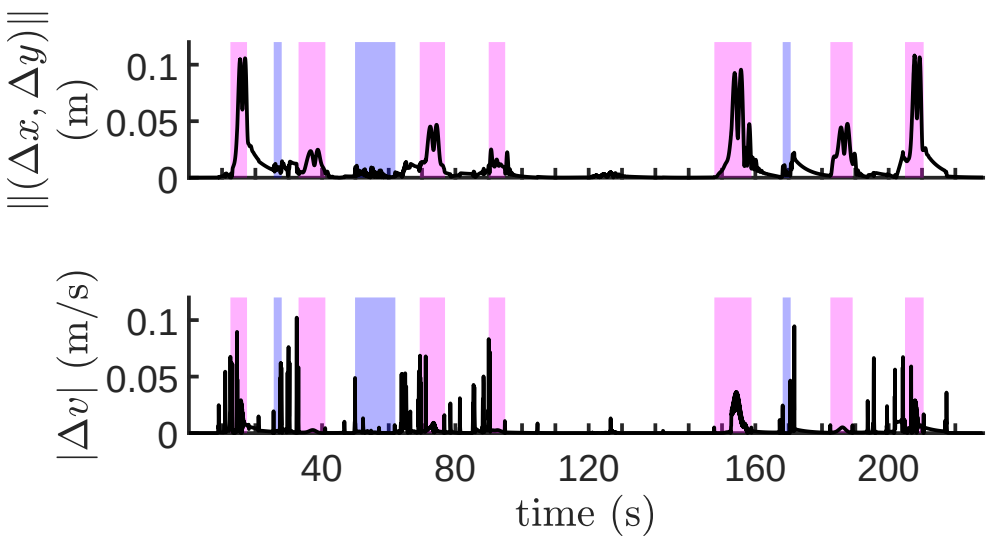



Figure 7.22: Sample tracking error. Pink bands indicate cornering maneuvers and blue bands indicate lane change maneuvers in the reference trajectory.

on the engine speed approximately equal to 2700rpm.

Because of the implicit switch introduced in the model we could not compare our solution to a BONMIN solution since we could not access the solver at the level of its iterates in order to utilize the symSS modification for hybrid systems (see: Section 7.4).

Further, for a wide swath of parameter values we could not find ones in which the C&R method was successful in simulation. This is likely due to the coupled nature of the controls. In this problem the binary control is not input directly but is multiplied by a scaled continuous control to yield the brake and throttle inputs. The resulting Hessian with relaxed binary input showed strong coupling between the controls $b, \theta$ that ultimately led to issues of numerical ill-conditioning of the Newton steps.

In Figs. 7.22 and 7.23 we display the error and controls of a sample simulation for CWQT with $s = 3, l = 2, r = 3$.

In Figs. 7.24 and 7.25 we display the position tracking performance of the CWQT and IQT methods, respectively, over various parameters. The upper limit of 0.5m is a threshold meaning some points coloured dark red may have a value even higher. For CWQT there is a large range of parameter values that give good performance whereas the IQT strategy has noticeable swaths of parameter values that yield undesirable performance. A value of

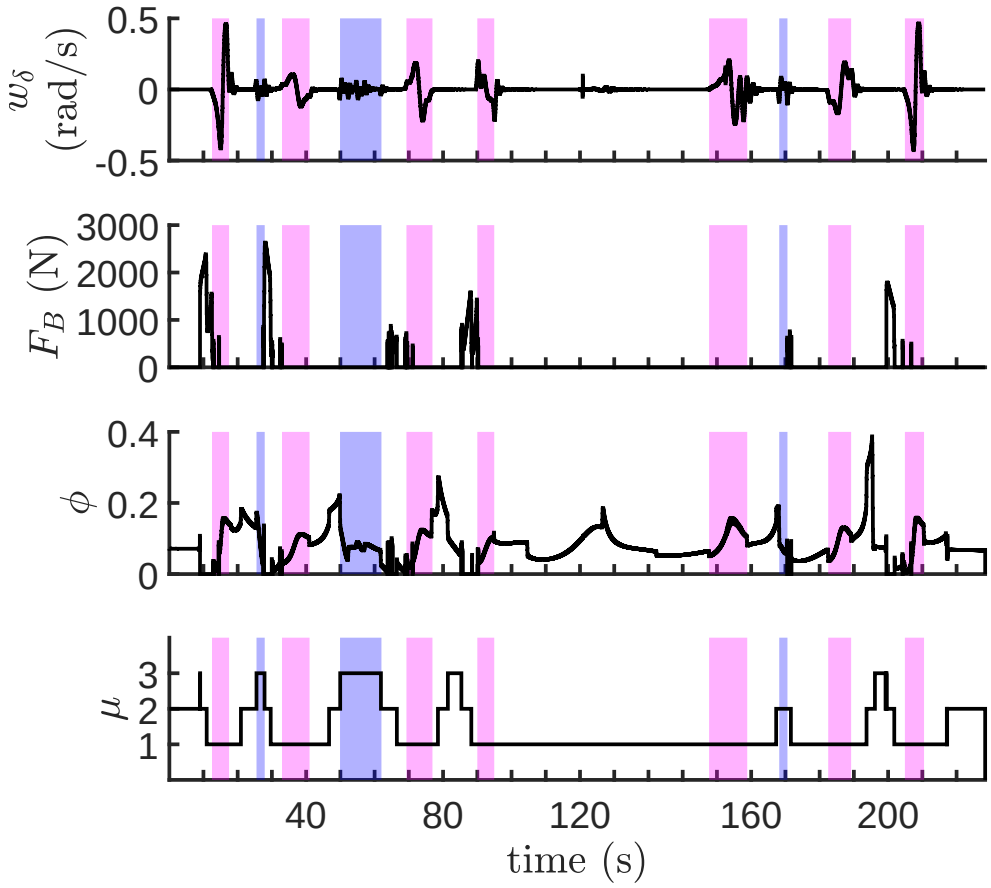



Figure 7.23: Sample control inputs. The gear $\mu$ is implicitly determined.

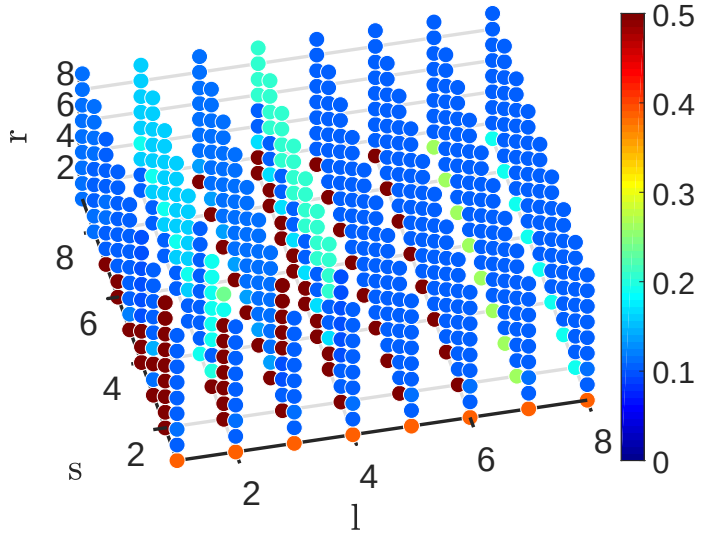



Figure 7.24: CWQT position tracking performance.

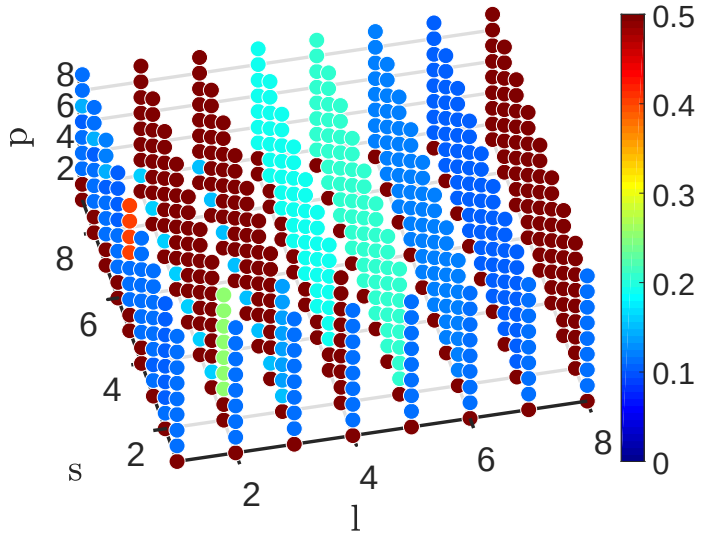



Figure 7.25: IQT position tracking performance.

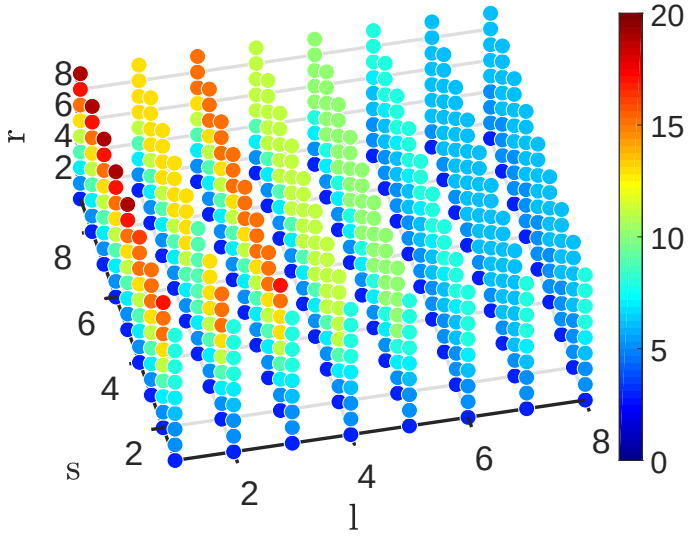



Figure 7.26: CWQT maximum number of NLP subproblems.

$r = 1$ or $p = 1$ introduces some performance degradation over many $s, l$ values for both strategies. Unlike the results of Section 7.5.1 there is no consistent trend in $l$.

In Figs. 7.26 and 7.27 we display the maximal number of NLP subproblems per timestep over various parameters. From these we can see the scaling of the CWQT problem size with $r$ and the scaling of IQT with $s$ as noted in Section 7.3. Similar to other results $l$ significantly shrinks the number of subproblems for larger values.

From this data the CWQT strategy appears to operate well over a wider range of values than IQT. It also appears that good performance can be achieved, in both cases, without requiring parameters that allow for a large number of subproblems.

## 7.6 Discussion

We have presented a strategy that relies on simple assumptions to handle the difficulties arising from discrete control inputs. This strategy, coupled with a greedy search method, provides a hard upper bound on computational complexity that makes it suitable for real-world problems with hard real-time constraints.

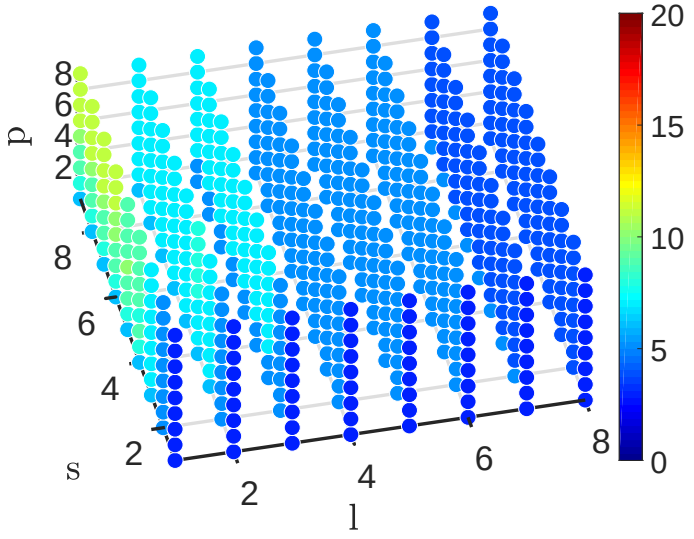



Figure 7.27: IQT maximum number of NLP subproblems.

By investigating a number of test problems, some with internal switches, we were able to demonstrate the practicality of the QT strategy. In terms of controller performance, the QT strategy performs favourably as compared to existing strategies, with a marked reduction in controller TATs — often on the order of an order of magnitude or more. In more than one case the QT strategy was able to find viable solutions where other solution methods simply failed.

In general, it was found that the CWQT strategy performed best when there was a significant difference in performance with the IQT strategy. It is the authors' opinion that future efforts should focus on the CWQT strategy and new variants thereof. The complexity of this strategy grows linearly in the number of permitted steps $r$ when using the greedy search strategy. We consistently found that small $r$ was sufficient for good performance meaning this strategy provides a very promising method to tightly bound the computational complexity of an online solver.

Theoretical questions raised by the QT strategy remain unexamined, such as, questions about controllability or controller stability. The QT strategy shares features with the Moving Window Blocking strategy presented in [55] which contains some stability and feasibility results for linear time invariant systems. However, it is expected that significant work is required to extend these results to a general hybrid nonlinear MPC setting with

QTs. At this stage, the QT strategy is a heuristic strategy that provides a possible method to handle the complexities of hybrid nonlinear systems in MPC while satisfying hard real-time constraints.

# Chapter 8

# Conclusions

Overcoming the hard real-time constraint of real world systems has proven to be a major challenge for MPC. In this thesis we present a variety of new strategies to accelerate the TATs of nonlinear and hybrid MPCs. These range from utilizing the power of symbolic computing for exact derivative generation, to restricting the FHOCP Lagrangian to an affine subspace, to truncating the FHOCP Hessian using a perturbative expansion, to enforcing common sense constraints on integer-controls. All of these work to reduce the computational burden of some component of the FHOCP solver whether it be derivative evaluation, a linear solve or integer constraint handling. There is a tradeoff to these strategies, in that the TAT acceleration comes at the potential cost of reduced controller performance. However, the acceleration in TAT yielded by some of these methods may be enough to meet the real-time constraint of the application, thus expanding the possibilities of MPC application.

The symSS formulation of a FHOCP leverages the power of symbolic computing to generate optimized code for exact derivative evaluation. This formulation can be applied with any derivative-based NLP. The purpose of this method is to handle the dynamic constraints, in particular, so that the dimension of the resulting NLP is of minimal dimension. We suggest that inequality constraints be handled with penalty functions when using this formulation, to maintain the minimal size of the problem. We used Maple in this thesis as it gave us the advantage of generating the computational sequence symbolically, carrying out procedural optimization, conducting automatic differentiation and generating optimized MATLAB or C code entirely within a single application. This kept the MPC development workflow efficient. A drawback to the symSS approach is scalability. Large models and large horizons may present an issue in generating the derivatives. However, as the code generation is done entirely offline, one can simply dedicate more computational resources

to the problem. At the current time, we can compute the derivatives of state-of-the-art problems with horizon lengths of 20-30 timesteps using nonlinear models of 10 states or less on a standard desktop. A further difficulty presented with this approach is the numerical ill-conditioning that can arise from using penalty functions. However, we note the issue of ill-conditioning is an issue for any inequality constrained problem and must ultimately be dealt with by the solver.

We introduced a novel application of POD to MPC reduction, called PODrMPC. Instead of using POD to reduce the order of the plant model, we instead applied POD to the Lagrangian of the FHOCP within NMPC. This allows for a greater reduction of the dimension of the optimization problem, leading to faster controller turnaround times. It was found that for sufficiently large reduction in dimension we can replace the linear solver with a fast symbolic form that further accelerates the TAT. A drawback of this method is that the degree of dimensional reduction is not analytically known. Further analysis must be done to go beyond the current common-sense approach of applying this method. Can we determine what constitutes a good snapshot? And how does this choice ultimately relate to the reduced dimension $r$ achieved? Even without being able to answer these questions it appears that the method is quite robust and can be applied to a wide variety of problems. However, it is difficult to provide, like with many nonlinear model reduction methods, any guarantee of performance.

There remain many unanswered theoretical questions about this method. Some are fundamental, such as, does PODrMPC preserve stability, controllability or feasibility of an MPC? Others are more numerics-oriented, like, how does the integration of POD reduced Newton steps affect the behaviour of more complex NLP methods like interior-point or sequential quadratic programming? At the level of the FHOCP it is clear that PODrMPC preserves convexity, since convexity is preserved under projection. There is a straightforward way to answer these questions, which is to remove the error of the method — entirely turning it into another exact solver — meaning all properties of the original MPC are retained. This idea was only suggested recently and has not been investigated. Since the error of the PODrMPC method comes from the fact we restrict our optimization problem to a subspace, we can eliminate this error by including a gradient step in the direction orthogonal to the subspace. This would incur minimal added computational cost but could eliminate entirely the error of the method, turning the POD reduced solver into an exact solver. The inclusion of orthogonal gradient steps could also increase the robustness of the method to the choice of snapshot data and possibly allow for greater dimensional reduction of the Newton step. The restricted Lagrangian approach reduces the linear solve cost in a Newton step. The cost of computing the full Hessian still remains. Some options that have not yet been investigated would be to utilize a first-order Hessian approximation scheme,

like the Broyden Fletcher Goldfarb Shanno algorithm, or some other cheap quasi-Newton method online along with the subspace restriction. In the offline stage we would still use the full Hessian with a robust NLP method to determine the subspace. This would reduce the convergence rate but also decrease the cost per iteration. The question to answer is: are more cheaper iterations faster than fewer expensive full Hessian iterations?

The IMBrMPC scheme can be seen as an extension of PODrMPC in which we have injected sparsity into the problem using the well-known strategy of move blocking. Because of the sparsity this method introduces, even greater TAT acceleration was found compared to the PODrMPC method for low dimension. The inclusion of orthogonal gradient steps could also be applied to this method if done with care. This method has many of the same unanswered questions as PODrMPC.

The truncated Lagrangian method takes advantage of the symSS formulation to yield a new quasi-Newton method for FHOCP solvers. This method relies on the balance that an MPC has to have between the plant dynamics and sampling frequency. For nonlinear problems where we can identify the timestep as a 'small' parameter this method is most appropriate. We also expect to achieve the greatest level of reduction for those models with a high degree of computational complexity. Computational complexity here is in terms of the form of statements making up the plant model expression and is not related to the dimension of the model. For models with simple expressions and fast dynamics in comparison to the timestep, this method will most likely not lead to desirable results. The truncated Lagrangian method provides a way to tackle the computational cost of the Hessian directly at the symbolic level. Theoretical questions about this quasi-Newton method remain, such as what is its convergence rate? For such an analysis we should begin with linear systems in order to relate the truncation with the eigenvalues of the model. In this way we can get an initial relationship between the timestep size and convergence properties.

We leverage the symSS to propose a nested MPC design for controller integration. This is a well known and difficult problem. The nested design we propose contains a controlled plant model in the outer layer that contains the actual lower level controller. The controlled plant model is in explicit form so we are able to generate derivatives to solve the outer MPC in real time. We focus on planning and controller integration for autonomous vehicles but this approach can be generalized to other integration problems where a controlled plant model is desired. Future work should focus on refining this approach and demonstrating its utility in special cases where pure replanning fails. This nested MPC approach is in a very early stage of development.

With the success of the PODrMPC method we hope to improve the nested MPC de-

sign by incorporating the PODrMPC in the controlled plant model. This will yield better predictions of the controlled plant and thus better overall performance. A drawback of this approach is scalability as the size of the controlled plant model can be very large even with our symSS strategy, making it very challenging to compute derivatives. However, dedicating more computing resources offline can alleviate some of this difficulty. Another option would be to turn to alternate problem formulations for the outer loop, like direct collocation. Issues of stability as well as other theoretical questions have not been examined. We would like to answer some of these questions in future works, and derive a stabilizing formulation of the outer MPC using a Lyapunov analysis of the controlled plant model. For example, if one has controllability of a bicycle, we can stabilize it by providing a straight line constant velocity path. It would be interesting to see if we can generate a stable system even if the low-level controller is unstable.

Lastly, the QT strategy presented provides a fast method to handle integer controls in hybrid MPC problems. This strategy enforces some degreee of continuity between timesteps of an MPC that has practical advantages. For integer controls, utilizing the optimum of an FHOCP from one timestep to the next is not necessarily desirable as chatter can be introduced to the input signal. One can view the QT strategy as a modification of the receding horizon principle for integer controls. Instead of implementing only $\mathbf{u}^*(0)$ at timestep $t_i$ we implement the subsequence $\mathbf{u}^*(0), \mathbf{u}^*(1), \ldots, \mathbf{u}^*(\tilde{n})$ of the solution at $t_i$ over the next $\tilde{n}+1$ timesteps, where $\tilde{n}$ varies depending on the solution and QT parameters. This is done for each timestep meaning there is an enforced overlap in integer solutions over sequential timesteps.

From our initial studies it appears the CWQT strategy by in large performs better than the IQT strategy, and future work should focus on it and its variants. This method requires us to solve a number of NLP subproblems. It would be desirable if we could select the integer controls without solving the entire NLP for each subproblem. Can we replace it with a smaller problem? Is there some insight we can draw from local gradient data that will inform our decision? These options could dramatically reduce the computational cost further. Another simple method to decrease cost for problems with large horizons is to change the CWQT greedy search from a linear one to a bisection type one. This would reduce the order of the computational cost from $O(r)$ to $O(\log r)$. But as satisfactory results have been typically found for small $r$ this benefit may not be dramatic. A tradeoff of this strategy is the enlargement of the MPC tuning problem since the QT parameters must be selected to satisfy the control objectives.

# References


[1] L. Del Re, F. Allgöwer, L. Glielmo, C. Guardiola, and I. Kolmanovsky, "Automotive model predictive control," *Lecture Notes in Control and Information Science*, 2010.

[2] M. Huang, D. Liao-McPherson, S. Kim, K. Butts, and I. Kolmanovsky, "Toward real-time automotive model predictive control: A perspective from a diesel air path control development," in *2018 Annual American Control Conference (ACC)*, pp. 2425–2430, IEEE, 2018.

[3] S. Di Cairano and I. V. Kolmanovsky, "Real-time optimization and model predictive control for aerospace and automotive applications," in *2018 Annual American Control Conference (ACC)*, pp. 2392–2409, IEEE, 2018.

[4] C. W. Rowley, "Model reduction for fluids, using balanced proper orthogonal decomposition," *International Journal of Bifurcation and Chaos*, vol. 15, no. 03, pp. 997–1013, 2005.

[5] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 22, no. 2, pp. 155–170, 2003.

[6] T. Ersal, H. K. Fathy, and J. L. Stein, "Realization-preserving structure and order reduction of nonlinear energetic system models using energy trajectory correlations," *Journal of Dynamic Systems, Measurement, and Control*, vol. 131, no. 3, p. 031004, 2009.

[7] L. S. Louca and B. U. Yildir, "Modelling and reduction techniques for studies of integrated hybrid vehicle systems," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 12, no. 2-3, pp. 203–218, 2006.

[8] C. Schmitke, K. Morency, and J. McPhee, "Using graph theory and symbolic computing to generate efficient models for multi-body vehicle dynamics," *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, vol. 222, no. 4, pp. 339–352, 2008.

[9] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[10] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control*, pp. 43–66, Springer, 2011.

[11] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology," in *AIche symposium series*, vol. 93, pp. 232–256, New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997.

[12] A. V. Rao, "Trajectory optimization: a survey," in *Optimization and optimal control in automotive systems*, pp. 3–21, Springer, 2014.

[13] L. T. Biegler, "An overview of simultaneous strategies for dynamic optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1043–1053, 2007.

[14] S. Kameswaran and L. T. Biegler, "Simultaneous dynamic optimization strategies: Recent advances and challenges," *Computers and Chemical Engineering*, vol. 30, no. 10, pp. 1560–1575, 2006.

[15] M. R. Garey and D. S. Johnson, *Computers and intractability*, vol. 29. wh freeman New York, 2002.

[16] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical programming*, vol. 39, no. 2, pp. 117–129, 1987.

[17] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*, pp. 391–417, Springer, 2009.

[18] C. Kirches, L. Wirsching, S. Sager, and H. G. Bock, "Efficient numerics for nonlinear model predictive control," in *Recent Advances in Optimization and its Applications in Engineering*, pp. 339–357, Springer, 2010.

[19] S. J. Wright and J. Nocedal, *Numerical optimization*, vol. 2. Springer New York, 1999.

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, pp. 1–36, 2018.

[22] S. J. Ohrem and C. Holden, "Modeling and nonlinear model predictive control of a subsea pump station," *IFAC-PapersOnLine*, vol. 50, no. 2, pp. 121–126, 2017.

[23] I. I. Sirmatel and N. Geroliminis, "Model predictive control of large-scale urban networks via perimeter control and route guidance actuation," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 6765–6770, IEEE, 2016.

[24] C. Eilers, J. Eschmann, and R. Menzenbach, "Underactuated trajectory-tracking control for long-exposure photography,"

[25] A. Albert, L. Imsland, and J. Haugen, "Numerical optimal control mixing collocation with single shooting: A case study," *IFAC-PapersOnLine*, vol. 49, no. 7, pp. 290–295, 2016.

[26] M. Ellis and P. D. Christofides, "On closed-loop economic performance under lyapunov-based economic model predictive control," in *American Control Conference (ACC), 2016*, pp. 1778–1783, IEEE, 2016.

[27] G. Sánchez, M. Murillo, L. Genzelis, N. Deniz, and L. Giovanini, "Mpc for nonlinear systems: a comparative review of discretization methods," in *2017 XVII Workshop on Information Processing and Control (RPIC)*, pp. 1–6, IEEE, 2017.

[28] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[29] R. M. Neal and J. Zhang, "High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees," in *Feature Extraction*, pp. 265–296, Springer, 2006.

[30] H. Chen, D. L. Reuss, D. L. Hung, and V. Sick, "A practical guide for using proper orthogonal decomposition in engine research," *International Journal of Engine Research*, vol. 14, no. 4, pp. 307–319, 2013.

[31] J. A. Atwell and B. B. King, "Reduced order controllers for spatially distributed systems via proper orthogonal decomposition," *SIAM Journal on Scientific Computing*, vol. 26, no. 1, pp. 128–151, 2004.

[32] P. Prabhat, S. Balakrishnan, D. C. Look, and R. Padhi, "Proper orthogonal decomposition based modeling and experimental implementation of a neurocontroller for a heat diffusion system," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2652–2657, IEEE, 2003.

[33] A. Marquez, J. J. Oviedo, D. Odloak, *et al.*, "Model reduction using proper orthogonal decomposition and predictive control of distributed reactor system," *Journal of Control Science and Engineering*, vol. 2013, p. 3, 2013.

[34] R. Masoudi, T. Uchida, and J. McPhee, "Reduction of multibody dynamic models in automotive systems using the proper orthogonal decomposition," *Journal of Computational and Nonlinear Dynamics*, vol. 10, no. 3, p. 031007, 2015.

[35] A. Alla and S. Volkwein, "Asymptotic stability of POD based model predictive control for a semilinear parabolic PDE," *Advances in Computational Mathematics*, vol. 41, no. 5, pp. 1073–1102, 2015.

[36] S. Volkwein, "Proper orthogonal decomposition: applications in optimization and control," *CEA-EDFINRIA Numerical Analysis Summer School*, 2007.

[37] K. Kunisch and S. Volkwein, "Optimal snapshot location for computing POD basis functions," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 44, no. 3, pp. 509–529, 2010.

[38] R. Quirynen, M. Vukov, and M. Diehl, "Multiple shooting in a microsecond," in *Multiple Shooting and Time Domain Decomposition Methods*, pp. 183–201, Springer, 2015.

[39] J. Andersson, *A general-purpose software framework for dynamic optimization*. PhD thesis, PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, 2013.

[40] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, "Auto-generated algorithms for nonlinear model predictive control on long and on short horizons," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 5113–5118, IEEE, 2013.

[41] G. Frison, H. B. Sørensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *Control Conference (ECC), 2014 European*, pp. 128–133, IEEE, 2014.

[42] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 289–329, 2015.

[43] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 668–674, IEEE, 2012.

[44] M. Rathinam and L. R. Petzold, "A new look at proper orthogonal decomposition," *SIAM Journal on Numerical Analysis*, vol. 41, no. 5, pp. 1893–1925, 2003.

[45] A. Maitland and J. McPhee, "Improving model predictive controller turnaround time using restricted Lagrangians," in *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pp. 3811–3816, IEEE, 2017.

[46] A. Maitland, M. Batra, and J. McPhee, "Nonlinear model predictive control reduction using truncated single shooting," in *2018 Annual American Control Conference (ACC)*, pp. 3165–3170, IEEE, 2018.

[47] S. Lall, P. Krysl, and J. E. Marsden, "Structure-preserving model reduction for mechanical systems," *Physica D: Nonlinear Phenomena*, vol. 184, no. 1, pp. 304–318, 2003.

[48] R. C. Thompson, "Principal submatrices IX: Interlacing inequalities for singular values of submatrices," *Linear Algebra and its Applications*, vol. 5, no. 1, pp. 1–12, 1972.

[49] H. Weyl, "Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung)," *Mathematische Annalen*, vol. 71, no. 4, pp. 441–479, 1912.

[50] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *The quarterly journal of mathematics*, vol. 11, no. 1, pp. 50–59, 1960.

[51] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[52] P. Karamanakos, T. Geyer, and S. Manias, "Direct voltage control of DC–DC boost converters using enumeration-based model predictive control," *IEEE Transactions on Power Electronics*, vol. 29, no. 2, pp. 968–978, 2014.

[53] R. Choroszucha, J. Sun, and K. Butts, "Nonlinear model order reduction for predictive control of the diesel engine airpath," in *American Control Conference (ACC), 2016*, pp. 5081–5086, IEEE, 2016.

[54] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves, "Model predictive control for the operation of building cooling systems," *IEEE Transactions on control systems technology*, vol. 20, no. 3, pp. 796–803, 2012.

[55] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *Journal of Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

[56] R. Gondhalekar and J.-i. Imura, "Recursive feasibility guarantees in move-blocking mpc," in *Decision and Control, 2007 46th IEEE Conference on*, pp. 1374–1379, IEEE, 2007.

[57] R. Gondhalekar, J.-i. Imura, and K. Kashima, "Controlled invariant feasibilitya general approach to enforcing strong feasibility in mpc applied to move-blocking," *Automatica*, vol. 45, no. 12, pp. 2869–2875, 2009.

[58] R. C. Shekhar and C. Manzie, "Optimal move blocking strategies for model predictive control," *Automatica*, vol. 61, pp. 27–34, 2015.

[59] R. C. Shekhar and J. M. Maciejowski, "Robust variable horizon mpc with move blocking," *Systems and Control Letters*, vol. 61, no. 4, pp. 587–594, 2012.

[60] A. Maitland and J. McPhee, "Towards integrated planning and control of autonomous vehicles using nested MPCs," in *ASME 2018 Dynamic Systems and Control Conference*, pp. V003T32A018–V003T32A018, American Society of Mechanical Engineers, 2018.

[61] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[62] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*, vol. 429. Springer Science & Business Media, 2012.

[63] I. Spangelo and O. Egeland, "Trajectory planning and collision avoidance for underwater vehicles using optimal control," *IEEE Journal of Oceanic Engineering*, vol. 19, no. 4, pp. 502–511, 1994.

[64] M. Werling and D. Liccardo, "Automatic collision avoidance using model-predictive online optimization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 6309–6314, IEEE, 2012.

[65] D. B. Leineweber, *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. PhD thesis, Université-it Heidelberg, 1998.

[66] I. M. Ross, P. Sekhavat, A. Fleming, and Q. Gong, "Optimal feedback control: foundations, examples, and experimental results for a new approach," *Journal of guidance control and dynamics*, vol. 31, no. 2, pp. 307–321, 2008.

[67] I. M. Ross, Q. Gong, F. Fahroo, and W. Kang, "Practical stabilization through real-time optimal control," in *American Control Conference, 2006*, pp. 6–pp, IEEE, 2006.

[68] M. Batra, J. McPhee, and N. L. Azad, "Parameter identification for a longitudinal dynamics model based on road tests of an electric vehicle," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. V003T01A026–V003T01A026, American Society of Mechanical Engineers, 2016.

[69] M. Batra, A. Maitland, J. McPhee, and N. L. Azad, "Non-linear model predictive anti-jerk cruise control for electric vehicles with slip-based constraints," in *2018 Annual American Control Conference (ACC)*, pp. 3915–3920, IEEE, 2018.

[70] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, 2014.

[71] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mechanical Systems and Signal Processing*, vol. 87, pp. 118–137, 2017.

[72] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," in *American Control Conference (ACC), 2017*, pp. 4023–4028, IEEE, 2017.

[73] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pp. 378–383, IEEE, 2013.

[74] M. Zanon, *Efficient Nonlinear Model Predictive Control Formulations for Economic Objectives with Aerospace and Automotive Applications*. PhD thesis, KU Leuven, 2015.

[75] M. Gerdts, "Solving mixed-integer optimal control problems by branch&bound: a case study from automobile test-driving with gear shift," *Optimal Control Applications and Methods*, vol. 26, no. 1, pp. 1–18, 2005.

[76] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear MPC in real-time," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 2505–2510, IEEE, 2014.

[77] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Control Conference (ECC), 2013 European*, pp. 4136–4141, IEEE, 2013.

[78] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[79] K. Berntorp and S. Di Cairano, "Joint decision making and motion planning for road vehicles using particle filtering," *IFAC-PapersOnLine*, vol. 49, no. 11, pp. 175–181, 2016.

[80] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, vol. 1, pp. 781–786, IEEE, 2006.

[81] T. Shim, G. Adireddy, and H. Yuan, "Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 226, no. 6, pp. 767–778, 2012.

[82] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles," *Control Engineering Practice*, vol. 17, no. 7, pp. 741–750, 2009.

[83] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *ASME 2010 dynamic systems and control conference*, pp. 265–272, American Society of Mechanical Engineers, 2010.

[84] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012.

[85] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *American Control Conference (ACC), 2012*, pp. 4239–4244, IEEE, 2012.

[86] J. hwan Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *American Control Conference (ACC), 2013*, pp. 188–193, IEEE, 2013.

[87] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[88] F. Zhu and P. J. Antsaklis, "Optimal control of hybrid switched systems: A brief survey," *Discrete Event Dynamic Systems*, vol. 25, no. 3, pp. 345–364, 2015.

[89] S. Srinivasan, I. Raptis, and E. R. Westervelt, "Low-dimensional sagittal plane model of normal human walking," *Journal of biomechanical engineering*, vol. 130, no. 5, p. 051017, 2008.

[90] R. Johansson and A. Rantzer, *Nonlinear and hybrid systems in automotive control*. Springer-Verlag New York, Inc., 2002.

[91] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, C. Pinello, and A. L. Sangiovanni-Vincentelli, "Automotive engine control and hybrid systems: Challenges and opportunities," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 888–912, 2000.

[92] E. F. Camacho, D. R. Ramírez, D. Limón, D. M. De La Peña, and T. Alamo, "Model predictive control techniques for hybrid systems," *Annual reviews in control*, vol. 34, no. 1, pp. 21–31, 2010.

[93] S. Sager, H. G. Bock, and G. Reinelt, "Direct methods with maximal lower bound for mixed-integer optimal control problems," *Mathematical Programming*, vol. 118, no. 1, pp. 109–149, 2009.

[94] H. G. Bock, C. Kirches, A. Meyer, and A. Potschka, "Numerical solution of optimal control problems with explicit and implicit switches," *Optimization Methods and Software*, vol. 33, no. 3, pp. 450–474, 2018.

[95] R. Jeroslow, "There cannot be any algorithm for integer programming with quadratic constraints," *Operations Research*, vol. 21, no. 1, pp. 221–224, 1973.

[96] I. E. Grossmann, "Review of nonlinear mixed-integer and disjunctive programming techniques," *Optimization and engineering*, vol. 3, no. 3, pp. 227–252, 2002.

[97] M. Schlüter, M. Gerdts, and J.-J. Rückmann, "A numerical study of MIDACO on 100 MINLP benchmarks," *Optimization*, vol. 61, no. 7, pp. 873–900, 2012.

[98] B. Borchers and J. E. Mitchell, "An improved branch and bound algorithm for mixed integer nonlinear programs," *Computers & Operations Research*, vol. 21, no. 4, pp. 359–367, 1994.

[99] I. Nowak, *Relaxation and decomposition methods for mixed integer nonlinear programming*, vol. 152. Springer Science & Business Media, 2006.

[100] C. Kirches, *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Springer, 2011.

[101] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, *et al.*, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optimization*, vol. 5, no. 2, pp. 186–204, 2008.

[102] C. Kirches, S. Sager, H. G. Bock, and J. P. Schlöder, "Time-optimal control of automobile test drives with gear shifts," *Optimal Control Applications and Methods*, vol. 31, no. 2, pp. 137–153, 2010.

[103] J. Currie, D. I. Wilson, *et al.*, "OPTI: lowering the barrier between open source optimizers and the industrial MATLAB user," *Foundations of computer-aided process operations*, vol. 24, p. 32, 2012.

[104] S. Sager, H. G. Bock, M. Diehl, G. Reinelt, and J. P. Schloder, "Numerical methods for optimal control with binary control functions applied to a Lotka-Volterra type fishing problem," in *Recent Advances in Optimization*, pp. 269–289, Springer, 2006.

[105] O. Sundstrom and L. Guzzella, "A generic dynamic programming matlab function," in *Control Applications,(CCA) and Intelligent Control,(ISIC), 2009 IEEE*, pp. 1625–1630, IEEE, 2009.

[106] Q. Lin, R. Loxton, and K. L. Teo, "Optimal control of nonlinear switched systems: Computational methods and applications," *Journal of the Operations Research Society of China*, vol. 1, no. 3, pp. 275–311, 2013.

[107] V. Rehbock and L. Caccetta, "Two defence applications involving discrete valued optimal control," *ANZIAM journal*, vol. 44, pp. 33–54, 2008.

[108] K. L. Teo, V. Rehbock, and L. S. Jennings, "A new computational algorithm for functional inequality constrained optimization problems," *Automatica*, vol. 29, no. 3, pp. 789–792, 1993.

[109] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen, "Look-ahead control for heavy trucks to minimize trip time and fuel consumption," *Control Engineering Practice*, vol. 17, no. 2, pp. 245–254, 2009.

[110] S. Terwen, M. Back, and V. Krebs, "Predictive powertrain control for heavy duty trucks," *IFAC Proceedings Volumes*, vol. 37, no. 22, pp. 105–110, 2004.

[111] M. Gerdts, "A variable time transformation method for mixed-integer optimal control problems," *Optimal Control Applications and Methods*, vol. 27, no. 3, pp. 169–182, 2006.

[112] S. Sager, C. Kirches, and H. G. Bock, "Fast solution of periodic optimal control problems in automobile test-driving with gear shifts," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 1563–1568, IEEE, 2008.

[113] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle System Dynamics*, vol. 21, no. S1, pp. 1–18, 1992.

[114] M. Gerdts, "Optimal control of ordinary differential equations and differential-algebraic equations," *Habilitation, University of Bayreuth*, 2006.

# APPENDICES

# Appendix A

# Autonomous Vehicle Model

The following vehicle model was used as a test case for many of the methods presented in the thesis. We describe it in detail below.

The vehicle model is a single-track nonlinear model with Pacejka tires to capture nonlinear slip dynamics [113]. This captures the planar motion of a vehicle assuming the pitch, heave, and roll are negligible and is a typical vehicle model used in controller design. The model also contains mappings relating the throttle and brake force to wheel torques. The control inputs $\mathbf{u} \in \mathbb{R}^3$ are steering angle rate $w_\delta \in [-0.5, 0.5]$ (rad/s), total braking force $F_B \in [0, 15000]$ (N) and throttle $\phi \in [0, 1]$. The states $\mathbf{x} \in \mathbb{R}^7$ are position $(x, y)$, speed $v$, side slip angle $\beta$, yaw angle $\psi$, rate of change of yaw $w_z$ and steering angle $\delta$. The values $x, y, v$ are taken at the center of gravity of the vehicle. We denote the state and control vectors by $\mathbf{x} = [x, y, v, \beta, \psi, w_z, \delta]^T$ and $\mathbf{u} = [w_\delta, F_B, \phi]^T$, respectively.

The model's equations of motion are given by an explicit system of differential equations

$$
\begin{aligned}
\dot{x} &= v\cos(\psi - \beta) \\
\dot{y} &= v\sin(\psi - \beta) \\
\dot{v} &= \frac{1}{m}[(F_{lr} - F_{Ax})\cos\beta + F_{lf}\cos(\delta + \beta) - (F_{sr} - F_{Ay})\sin\beta - F_{sf}\sin(\delta + \beta)] \\
\dot{\beta} &= w_z - \frac{1}{mv}[(F_{lr} - F_{Ax})\sin\beta + F_{lf}\sin(\delta + \beta) + (F_{sr} - F_{Ay})\cos\beta + F_{sf}\cos(\delta + \beta)] \\
\dot{\psi} &= w_z \\
\dot{w}_z &= \frac{1}{I_{zz}}\left(F_{sf}l_f\cos\delta - F_{sr}l_r - F_{Ay}e_{SP} + F_{lf}l_f\sin\delta\right) \\
\dot{\delta} &= w_\delta
\end{aligned}
$$

where the forces are given by

$$
\begin{aligned}
F_{sf} &= D_f \sin(C_f \arctan(B_f\alpha_f - E_f(B_f\alpha_f - \arctan(B_f\alpha_f)))) \\
F_{sr} &= D_r \sin(C_r \arctan(B_r\alpha_r - E_r(B_r\alpha_r - \arctan(B_r\alpha_r)))) \\
F_{Ax} &= \frac{1}{2} c_W \rho A v^2 + D_x \\
F_{Ay} &= D_y \\
F_{lf} &= -\frac{2}{3} F_B - f_R \frac{m l_r g}{l_f + l_r} \\
F_{lr} &= \frac{\zeta M_{mot}}{R} - \frac{1}{3} F_B - f_R \frac{m l_f g}{l_f + l_r}
\end{aligned}
$$

and the intermediate terms

$$
\begin{aligned}
\alpha_f &= \delta - \arctan\left(\frac{l_f w_z - v \sin\beta}{v \cos\beta}\right) \\
\alpha_r &= \arctan\left(\frac{l_r w_z + v \sin\beta}{v \cos\beta}\right) \\
M_{mot} &= (1 - e^{-3\phi})\left(-37.8 + 1.54\frac{\zeta v}{R} - 0.0019\left(\frac{\zeta v}{R}\right)^2\right) + e^{-3\phi}\left(-34.9 - 0.04775\frac{\zeta v}{R}\right) \\
f_R &= 9 \times 10^{-3} + 7.2 \times 10^{-5} v + 5.038848 \times 10^{-10} v^4
\end{aligned}
$$

are the front slip angle, rear slip angle, motor torque and speed-dependent rolling resistance, respectively. We display a schematic of the model in Fig. A.1. The parameter $\zeta = i_g(\mu) i_t$ is a function of the gear input $\mu \in \{1, 2, 3, 4, 5\}$. For most cases $\mu$ is fixed to yield a model with entirely continuous inputs. The model parameters are: $m$ car mass, $g$ gravitational acceleration, $l_f, l_r, e_{SP}$ vehicle dimensions (see: Fig. A.1), $R$ wheel radius, $I_{zz}$ car moment of inertia, $c_W$ air drag coefficient, $\rho$ air density, $A$ effective flow surface, $i_t$ motor torque transmission and $B_{f,r}, C_{f,r}, D_{f,r}, E_{f,r}$ Pacejka tire model parameters. Furthermore, the terms $D_x, D_y$ are forces on the body of the vehicle coming from environmental disturbances. These are both zero if we simulate the model in the nominal case. We note that the equations of motion of this model are given by ordinary differential equations and so we can express the model compactly by $\dot{\mathbf{x}} = \Phi(\mathbf{x}, \mathbf{u})$. More model details and parameter values can be found in [75, 114, 102].

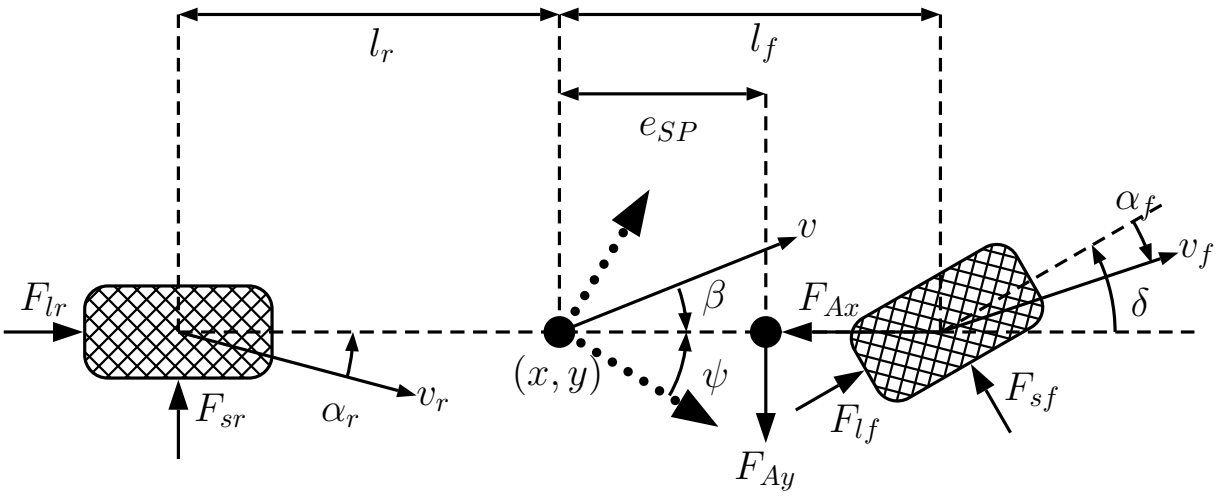



Figure A.1: Vehicle model schematic.

The Jacobians of the model used in Section 4.2.4 are given below:

$$A(\bar{\mathbf{x}}_0, \bar{\mathbf{u}}_0) = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -v_0 & v_0 & 0 \\ 0 & 0 & A_{33} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{54} & A_{55} & 0 & A_{57} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & A_{74} & A_{75} & 0 & A_{77} \end{pmatrix}$$

$$B(\bar{\mathbf{x}}_0, \bar{\mathbf{u}}_0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1/m & B_{33} \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$
\begin{aligned}
A_{33} &= f_R'(v_0)g - \frac{c_W \rho A v_0}{m} + \frac{1}{R^3}\frac{M_1'(v_0)e^{-3\phi_0} - M_2'(v_0)}{m} \\
A_{54} &= -\frac{D_f C_f B_f}{m v_0} + f_R(v_0) g \frac{1}{v_0}\frac{m l_r}{(l_f + l_r)} \\
A_{55} &= -\frac{1}{R^3}\left(\frac{D_f C_f B_f}{m v_0} + \frac{D_r C_r B_r}{m v_0}\right) + f_R(v_0) g \frac{1}{v_0} \\
&\quad + \frac{1}{2}\frac{c_W \rho A v_0}{m} + \frac{1}{v_0}\frac{M_1(v_0)e^{-3\phi_0}}{m} + \frac{1}{R^3}\frac{1}{v_0}\frac{M_2(v_0)}{m} \\
A_{57} &= \frac{l_f}{v_0}\frac{D_f C_f B_f}{m v_0} - \frac{l_r}{v_0}\frac{D_r C_r B_r}{m v_0} + 1 \\
A_{74} &= \frac{l_f}{I_{zz}} D_f C_f B_f - f_R(v_0) g \frac{l_f}{I_{zz}}\frac{m l_r}{(l_f + l_r)} \\
A_{75} &= \frac{l_f}{I_{zz}} D_f C_f B_f - \frac{l_r}{I_{zz}} D_r C_r B_r \\
A_{77} &= -\frac{l_f}{v_0}\frac{l_f}{I_{zz}} D_f C_f B_f - \frac{l_r}{v_0}\frac{l_r}{I_{zz}} D_r C_r B_r \\
B_{33} &= 3\frac{1}{R^3}\frac{M_1(v_0)e^{-3\phi_0}}{m}
\end{aligned}
$$

and

$$
\begin{aligned}
f_R(v_0) &= 0.009 + 0.000072 v_0 + 5.038848 \times 10^{-10} v_0^4 \\
M_1(v_0) &= -0.0019 v_0^2 \zeta^3 + 1.58775 v_0 R \zeta^2 - 2.9 R^2 \zeta \\
M_2(v_0) &= 37.8 R^2 \zeta - 1.54 v_0 R \zeta^2 + 0.0019 v_0^2 \zeta^3
\end{aligned}
$$

The linearization points are $\bar{\mathbf{x}}_0 = [0, 0, v_0, 0, 0, 0, 0]^T$, $\bar{\mathbf{u}}_0 = [0, 0, \phi_0]^T$.